BUILDING A PREDICTIVE MODEL ON STATE OF GOOD REPAIR BY MACHINE LEARNING ALGORITHM ON PUBLIC TRANSPORTATION ROLLING STOCK

A Dissertation
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Dilip Kumar Mistry

In Partial Fulfillment of the Requirements
for the Degree of
DOCTOR OF PHILOSOPHY

Major Program:
Transportation and Logistics

May 2018

Fargo, North Dakota

North Dakota State University

Graduate School

Title

BUILDING A PREDICTIVE MODEL ON STATE OF GOOD REPAIR

BY MACHINE LEARNING ALGORITHM ON PUBLIC

TRANSPORTATION ROLLING STOCK

By

Dilip Kumar Mistry

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

DOCTOR OF PHILOSOPHY

SUPERVISORY COMMITTEE:

Dr. Jill Hough
Chair

Dr. Bruce Maylath

Dr. Alan Dybing

Dr. Michal Jaroszynski

Approved:

July 2, 2018
Date

Joseph Szmerekovsky
Department Chair

ABSTRACT

Achieving and maintaining public transportation rolling stocks in a state of good repair is very crucial to provide safe and reliable services to riders. Besides, transit agencies who seek federal grants must keep their transit assets in a state of good repair. Therefore, transit agencies need an intelligent predictive model for analyzing their transportation rolling stocks, finding out the current condition, and predicting when they need to be replaced or rehabilitated. Since many transit agencies do not have good analytical tools for predicting the service life of vehicles, this simple predictive model would be a valuable resource for their state of good repair needs and their prioritization of capital needs for replacement and rehabilitation.

The ability to accurately predict the service life of revenue vehicles is crucial achieving the state of good repair. In this dissertation, three unique tree-based ensemble learning methods have been applied to build three predictive models. The machine learning methods used in this dissertation are random forest regression, gradient boosting regression, and decision tree regression. After evaluation and comparison of the performance results amongst all models, the gradient boosting regression model with the top 30 most important features was found to be the best fit for predicting the service life of transit vehicles. This model can be used to predict the projected retired year for all nationwide vehicles in operation, the single transit agency's transit vehicle, and any single vehicle.

The revenue vehicle inventory data from National Transit Database (NTD) has been used to build the machine learning predictive model. Before feeding the data into the model, a variety of new features were created, missing data were fixed, and extreme values or outliers were handled for the machine learning algorithm.

ACKNOWLEDGMENTS

I would like to thank to my entire family, and especially my beloved wife, Lupa Mistry. Thank you for supporting me for everything, and especially I can't thank you enough for encouraging me throughout this experience. To my beloved daughter Authoi Mistry, I would like to express my thanks for being such a good girl. Without your loving support, I could not have achieved this important goal.

I would like to thank my advisor Dr. Jill Hough for her guidance throughout my time at North Dakota State University, Fargo, North Dakota. Your advice throughout the development of my dissertation as well as on my career has been invaluable. You have inspired me in many difficult situations. You have served well as my guide, not just in my academics, but in life in general. You have been a tremendous mentor for me. You have been very patient with my shortcomings and have always encouraged me to grow as a research scientist. Your advice was very motivational and gave me a new perspective to achieve my dream. I sincerely appreciate your valuable time and I will stay forever grateful.

I am also grateful to my supervisory committee and would like to thank Dr. Bruce Maylath, Dr. Alan Dybing, and Dr. Michal Jaroszynski for serving as my committee members. I also want to thank you for letting my defense be an enjoyable moment, and for your brilliant comments and suggestions.

Finally, I would like to thank NDSU Graduate Center for Writers, especially Kristina Caton for proofreading my dissertation. My dissertation would not have been complete without the aid and support of Graduate Center for Writers. During the proofreading period, Kristina always gave me much encouragement and sound advice.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AAAPUL	Average Age of Assets as a Percentage of their Useful Life.
AASHTO	American Association of State Highway and Transportation Officials.
ADA	Americans with Disability Act of 1990.
APTA	American Public Transportation Association.
Bagging	Bootstrapped Aggregation.
BART	San Francisco Bay Area Rapid Transit.
C & P	Conditions and Performance.
CIP	Capital Investment Program.
CIS	Capital Investment Strategy.
CSV	Comma Separated Values.
CTA	Chicago Transit Authority.
CTAMS	CalTrain Asset Management System.
DO	Directly-Operated.
DOAV	Department of Aviation.
DOT	Department of Transportation.
DRPT	Department of Rail and Public Transit.
DTR	Decision Tree Regression.
EAM	Enterprise Asset Management.
FAST Act	Fixing America's Surface Transportation Act.
FHWA	Federal Highway Administration.
FRA	Federal Railroad Administration.
FTA	Federal Transit Administration.
GAO	Government Accountability Office.

GBM .....Gradient Boosting Machines.

GBR .....Gradient Boosting Regression.

GBRT .....Gradient Boosted Regression Trees.

JPB .....Peninsula Corridor Joint Powers Board.

LCARE .....Life Cycle Asset Rehabilitation Enhancement.

MAE.....Mean Absolute Error.

MAP-21.....Moving Ahead for Progress in the 21st Century.

MARTA .....Metropolitan Atlanta Rapid Transit Authority.

MAT Bus .....Fargo Metropolitan Area Transit.

MBTA .....Massachusetts Bay Transportation Authority.

MMIS .....Maintenance Management Information System.

MPO .....Metropolitan Planning Organizations.

MTA.....Metropolitan Transportation Authority of New York.

MTC .....Metropolitan Transportation Commission.

NCHRP .....National Cooperative Highway Research Program.

NHTS .....National Household Travel Survey.

NJDOT .....New Jersey Department of Transportation.

NJT.....New Jersey Transit.

NJTA.....New Jersey Turnpike Authority.

NJ Transit.....New Jersey Transit Corporation.

NTD .....National Transit Database.

NYCT.....New York City Transit.

OOB .....Outside of the Bag.

PI.....Prioritization Index.

PROGGRES.....Program Guidance and Grant Evaluation System.

PT.....Purchased Transportation.

RFR.....Random Forest Regression.

RMSE.....Root Mean Squared Error.

RTA.....Regional Transit Authority.

RTCI .....Regional Transit Capital Investment.

RVI ID .....Revenue Vehicle Inventory ID.

SAS .....Statistical Analysis System.

SEPTA .....Southeastern Pennsylvania Transportation Authority.

SJTA .....South New Jersey Transportation Authority.

SGR.....State of Good Repair.

SOGR.....State of Good Repair.

TAM.....Transit Asset Management.

TAMP .....Transit Asset Management Plan.

TAPT.....Transit Asset Prioritization Tool.

TCRP.....Transit Cooperative Research Program.

TERM .....Transit Economic Requirements Model.

Trans-AM.....Transit Asset Management System.

TRB.....Transportation Research Board.

U.S. DOT .....United States Department of Transportation.

VDOT .....Virginia Department of Transportation.

VPA.....Virginia Port Authority.

WMATA .....Washington Metropolitan Area Transit Authority.

LIST OF APPENDIX TABLES

CHAPTER 1. INTRODUCTION

The United States public transportation agencies are experiencing an increase in public transportation use and facing challenges maintaining their existing transit assets. These agencies have a variety of transit assets such as buses, trains, track, rights of way, facilities, and other assets in operation. Most of the transit assets have either aged or are beyond their recommended useful life. These assets need to be rehabilitated or replaced to maintain the state of good repair (SGR) to keep up with increased ridership. But due to lack of funding, the transit agencies expect their systems will suffer a significant reduction in service reliability, which will cause restricted transit services (Cambridge Systematics, 2009). Therefore, transit agencies need an intelligent predictive model that will help them to accurately predict when a transit asset needs to be rehabilitated and replaced; this will enable agencies to make decisions on investment and prioritize to maintain SGR needs.

1.1. Background

The “Moving Ahead for Progress in the 21st Century” or MAP21 law passed in 2012 was the Federal Transit Administration's (FTA) first and only standalone initiative for the state of good repair program. The MAP21 granted $2.14 billion in the fiscal year (FY) 2012 and $2.17 billion in the fiscal year 2013 for repairing and upgrading nation's transit rail and bus services to provide reliable, efficient, and safe services to riders (FTA, 2012). Then, the “Fixing America’s Surface Transportation Act” or FAST Act law passed in 2015 was built upon continuing most of MAP21’s provisions, along with other federal programs. The FTA estimated that there was about 25 percent of U.S. rail transit and 40 percent of buses that were in a marginal or poor condition in 2015. Therefore, the FTA prioritized maintaining bus and rail systems in a state of good

repair, so the FAST Act program increased annual funding from $2.1 billion to $2.5 billion for the FTA's state of good repair (5337) program (FTA, 2017).

Section 5326 of MAP21 requires the FTA to establish a definition for "State of Good Repair" that will have objective standards to measure the condition of various capital assets such as rolling stock, equipment, facilities, and infrastructure (Cevallos, 2016). However, there are no universal definitions of "State of Good Repair" adopted for public transit (Cohen & Barr, 2012). For example, according to "Transit Asset Management Practices," SGR is defined as "an asset or system is in a state of good repair when no backlog of capital needs exists – hence all asset life-cycle investment needs (e.g., preventive maintenance and rehabilitation) have been addressed and no capital asset exceeds its useful life" (FTA, 2010b, pp. Sec. 2-2). On the other hand, the American Public Transportation Association (APTA) Transit Asset Management Working Group defines SGR as "a condition in which assets are fit for the purpose for which they were intended" (APTA, 2013, p. 1). Both agency and its stakeholders accept this definition as it is comparatively easy. The Department of Transportation defined SGR as "a condition in which the existing physical assets, both individual and as a system, (a) are functioning within their 'useful lives,' and (b) are sustained through regular maintenance and replacement program" (Amtrak, 2009, p. 9). The state of good repair does not ensure the growth of service, but it provides a solid foundation so that transit agencies are reliable as ridership grows. Under normal conditions, certain transit assets reach at the end of their useful lives. These assets would be either replaced or renewed by creating annualized funding referred as normalized replacement cost. Since many transit assets have been deferred in the past and those assets have not been allotted funding for replacement, a significant backlog has been accumulated which is known as SOGR backlog (Amtrak, 2009).

Most transit agencies defined state of good repair in some ways so that they could keep their transit assets in ideal conditions. However, they defined it in the more of the same manner and concepts as (a) maintaining a transit agency's rolling stock and transit infrastructure in a specific level, (b) performing maintenance, repair, and rehabilitation, and (c) eliminating the agency's backlog (FTA, 2010b). The definition of SGR by transit agencies are listed below (FTA, 2008):

- The Massachusetts Bay Transportation Authority (MBTA) defines SGR as a standard where all transit assets are in an ideal condition within their design life.
- The New York City Transit Authority (NYCT) defines SGR as investments which cover depreciated asset conditions.
- The Southeastern Pennsylvania Transportation Authority (SEPTA) defines SGR in transit asset when no backlog exists, and each asset maintains its useful life. It also adjusts past deferred maintenance and replaces assets which exceed their useful life.
- The New Jersey Transit (NJT) defines SGR by achieving infrastructure components with replacing in scheduled maintenance within their life expectancy.
- The Cleveland Regional Transit Authority (RTA) defines SGR as a system where it maintains a consistent and high-quality condition system-wide.

1.2. Problem Statement

Public transportation is a critical transportation mode in the United States for a wide range of riders and is crucial to the nation's transportation system. There are about 1500 transit agencies in the United States that provide bus services and about 80 agencies that provide rail services. These transit agencies provide services to tens of millions of Americans every day, especially in large metropolitan areas. However, some of the major transit systems are more than

one hundred years old. Most of the transit assets are either suffering from underinvestment or lack of optimal transit asset management practices (APTA, 2007; US DOT, 2013). Thus, insufficient investment in capital transit assets is deteriorating much of the nation's transit assets. In addition, operating costs for maintaining transit assets beyond their original service expectancy are getting higher. This means that the reliability of service decreases as more transit assets breakdown during service. Overall, the quality of the stations and shelters, as well as the public safety, decline as aging assets fail to perform properly. As a result, the transit agencies become unreliable and less attractive for potential passengers (McCollom & Berrang, 2011). Furthermore, according to the FTA in its "National State of Good Repair Assessment," approximately one-third of the nation's transit assets are not in good shape. And, another analysis conducted with Transit Economic Requirements Model (TERM) on current physical and service condition shows that about one-third of the transit rail and bus are exceeding their useful life and reinvestments are needed to bring nation's transit vehicles to the state of good repair (FTA, 2008).

Concurrently, ridership of public transportation is increasing over time. For example, as reported by the APTA, public transportation ridership expanded by 34% from 1995 through 2012, which is higher than the 17% increase in the United States population over the same period (APTA, 2017). Another result from the National Household Travel Survey (NHTS) also shows that transit ridership increased by 16 percent from 2001 to 2009 which exceeds the population growth forecast during that period (FHWA, 2010). In addition is a report of increased ridership by the Metropolitan Planning Organizations (MPOs), who projects a low growth scenario for transit, predicting the overall ridership will grow 1.7 % per year from 2012 to 2032. In the same report by the MPOs, the high growth scenario is based on the historical trend of ridership over

the last 15 years and predicts that the future ridership will grow about 2.2 percent per year from 2012 to 2032. Both growth scenarios assess the level of investment needed for SGR. In fact, TERM estimates the average annual level of investment for the nation would be $24.5 billion, including $17.4 billion for replacing and rebuilding assets and $7.1 billion for expansion to keep up with ridership growth (FHWA, 2010).

In 2009, the FTA estimated that nearly $78 billion is needed to bring the nation's transit assets into a state of good repair (US GAO, 2013). The FTA used TERM to estimate normal replacement expenditures and estimated that an average of $14.4 billion per year was needed to maintain the state of good repair (FTA, 2010a). The FTA also calculated with the TERM model that an annual investment of $18.3 billion was needed to achieve a state of good repair over a 20-year period while maintaining the normal replacement needs. The potential consequences of keeping the above reinvestment rate suggests that the continued reinvestment may deteriorate the overall condition on the nation's transit assets and the rate of transit assets which already exceeded their useful life will increase to more than 30% by 2029 (FTA, 2010a). Moreover, if transit assets currently in acceptable condition are not replaced or rehabilitated on time, the transit service will result in increased operating costs, reduced safety, disrupted on-time service, and reduced ridership (US GAO, 2013). Therefore, it is imperative that the FTA and transit agencies look for ways to solve these critical issues in order to sustain the state of good repair.

However, in order to implement plans to sustain that state of good repair, the tools used to analyze the state of good repair must also be credible, reliable, and accurate. In 2010, the FTA assessed the accuracy of TERM's projections, which was published in '2004 Conditions and Performance' report. This assessment compared the projections made by the TERM model and the agencies' experiences. The comparison showed that the transit agencies total expenditures

were $10.5 billion from the year 2003 to 2009, whereas the TERM's projection was $12.1 billion. This comparison result indicated TERM was projecting 14 percent above the actual expenditure in the 2004 Condition & Performance report. The FTA also examined the condition rating reported by agencies over the period between 2003 and 2009. They found that if the actual agency expenditure is below the TERM's projections, the condition rating declines, whereas if the agency expenditure is above the TERM's projections, the condition rating improves (TRB, 2013). More specifically, in order to access the accuracy, the FTA's TERM tool forecasts the yearly replacement needs based on decay curves derived from data from selected transit systems. However, estimating an overall decay curve based on data from a single transit system, and then projecting replacement needs for all based on that single decay curve, may not correctly estimate the backlog by the TERM model because the decay curve is based on single transit system in a single transit environment.

A second assessment was performed by the FTA committee on the accuracy of the TERM's backlog estimation by comparing it with the 20-year capital spending requirements of three major agencies: MARTA, NYCT, and MBTA. The comparison results showed that the NYCT had 40% less than the TERM forecast for rail vehicles because they increased their rolling stock replacement age to 40 years while TERM kept it 28 to 29 years. It appears that the NYCT changed their backlog definition to a condition-based replacement criterion rather than an age-based replacement criterion. In addition, the MARTA found a discrepancy of $1.64 billion (52%) between the two forecasts in the revenue vehicle category. Again, this discrepancy is due to the difference in condition-based replacement rather than the TERM's age-based approach. Furthermore, the MBTA found a disparity of agency forecast of $0.53 billion (16%) higher than the TERM forecast in revenue vehicle category. This difference between these two forecasts is

due to the difference in the definitions of replacement conditions. Table 1 below shows the discrepancies between all three agencies' forecasts and the TERM forecast in the revenue vehicle category (Zarembski, 2013; TRB, 2013).

Table 1. TERM vs. Agencies' 20 Year Capital Needs Forecasts by Revenue Vehicle Category (in millions of $)

Category	NYCT			MARTA			MBTA		
	TERM	Agency	Diff	TERM	Agency	Diff	TERM	Agency	Diff
Revenue Vehicles	18,729	11,278	40%	3,127	1,488	52%	3398	3925	-16%

Source: Adapted from Zarembski, Allan M. 2013. *Analysis of Transit 20 Year Capital Forecasts: FTA TERM Model vs. Transit Estimates*. Washington, D.C.: Transportation Research Board of the National Academies. http://onlinepubs.trb.org/onlinepubs/reports/TERM_March_2013Zarembski.pdf.

The FTA committee recommended refining the TERM model and developing simple methods to project capital spending more accurately (TRB, 2013). Therefore, in order to reduce the discrepancies between the TERM forecast and agency forecasts, the FTA should reexamine its asset life criterion and should improve the TERM model to include a condition-based replacement approach (Zarembski, 2013). Thus, a simple predictive model is needed to supplement the TERM tool used to estimate the conditions of the revenue vehicles and project capital expenditure needs.

In conclusion, the problem statement of this research summarizes that the FTA's current TERM tool may have shortcomings in predicting the service life of transit vehicles. Furthermore, there is a problem with aging infrastructure, including the transit vehicles. Since the ridership in public transit has been increasing and is projected to grow, and in light of ongoing funding issues, both the FTA and transit agencies need an alternative way to accurately forecast the service life of transit vehicles. Therefore, this research involves building a predictive model by a machine learning algorithm that will more accurately predict the service life of transit vehicles

and perform statistical data analysis. This predictive tool will be useful to national, state, and local transit agencies, as well as researchers in transit asset management.

1.3. Objective

Maintaining transit rolling stocks in a state of good repair has become a strategic goal for transit agencies and the FTA. The challenge that transit agencies face maintaining assets in a state of good repair is that most agencies do not have an effective way to manage their physical transit assets (FTA, 2010b). The objective of this research is to develop a predictive model with machine learning algorithms for transit agencies to obtain a state of good repair so they can effectively prioritize capital investment for rehabilitation and replacement of transit vehicles. Although, transit agencies are aware of the consequences of the underinvestment of their assets, they have limited resources to predict the outcomes of various funding scenarios. Because the tools they currently use are not reliable, transit agencies cannot project accurate timelines for replacing assets when needed. These limitations prohibit transit agencies from addressing ongoing backlog replacement and rehabilitation issues when funding is insufficient (McCollom & Berrang, 2011). In order to address this problem, this research will develop a predictive model using machine learning techniques to help transit agencies to predict the service life of transit vehicles and calculate investment needs for rehabilitation and replacement needs of revenue vehicles. To do this, the research will investigate the transit state of good repair, asset management practices, fundamental concepts of transit asset management (TAM), and application of machine learning algorithms.

1.4. Organization

The research begins with the abstract that highlights the overall summary of the dissertation. The main thesis is organized into five chapters. In Chapter 1, the background of the

state of good repair problem, the problem statement, and the objectives of this research are discussed. In Chapter 2, the previous study on the state of good repair and asset management practices are presented. This chapter also includes early research on the state of good repair, an overview of the transit asset management system, the current condition of the United States transportation system, analytical tools used for the state of good repair, and a review of transit state of good repair practices in the United States. In Chapter 3, the methodology developed for service life prediction is presented, and three machine learning algorithms are introduced. Chapter 4 presents the proposed predictive models by building, evaluating, and comparing three regression models: gradient boosting regression, random forest regression, and decision tree regression. Chapter 4 also presents the preprocessing of data, data analysis, and challenges. Chapter 5 concludes the study and sets out goals for further research.

CHAPTER 2. LITERATURE REVIEW

This chapter provides an overview of the related literature on the state of good repair, existing asset management practices, and support tools that are currently used by many transit agencies and other relevant published articles. The review will focus on how the FTA maintains its current minimum service life policy, how transit authorities approach asset management, how they define and practice the state of good repair for transit assets, and how they identify the best practices to maintain the state of good repair. The sources of the literature review are from Federal Transit Agency (FTA) publications, Transportation Research Board (TRB) proceedings, Transit Cooperative Research Program (TCRP) publications, National Cooperative Highway Research Program (NCHRP) publications, and other published articles. The following key concepts from the above resources are summarized below.

2.1. Overview of Early Research on the State of Good Repair

The "NCHRP Report 545: Analytical Tools for Asset Management, Reviewed Asset Management Tools and Systems" published in 2005, provided two software tools, which are AssetManager NT and AssetManager PT (Cambridge Systematics, 2005). These tools were intended for the state departments of transportation and transit agencies to support tradeoff analysis for transportation asset management. The tools were developed to integrate with existing systems to help agencies to analyze and predict investment decisions for their transit assets. The report provided a snapshot of how existing tools were being used, what capabilities and limitations existed in the available asset management tools, and what kind of new tools were needed (Cambridge Systematics, 2005).

Then, the "NCHRP Report 551: Performance Measures and Targets for Transportation Asset Management" published in 2006, provided concepts of performance management for

transit agencies used for transit asset management (Cambridge Systematics, 2006). This report described how performance measures could be used for decision-making processes and resource optimization. It presented a framework for performance measure development and target values for use in asset management. In addition, it also provided best practices on how to set the performance target and what factors need to be considered when setting the performance target (Cambridge Systematics, 2006).

The "Useful Life of Transit Buses and Vans" research published in 2007 by the FTA assessed the policy on existing minimum service life for transit buses and vans (Laver, Schneck, Skorupski, & Cham, 2007). The study team interviewed transit agencies and performed engineering and economic analysis to evaluate the minimum service-life policy. The engineering analysis showed that the bus lifespan was restricted by the bus structure, while the economic analysis showed that the optimal replacement points for various bus types were at or later than the FTA's minimum service life. The study provided details on the useful life of buses and vans, the minimum service life policy by the FTA, the impact of the vehicle life expectancies, agency's decision on retirement, vehicle maintenance, and replacement best practices. The study also showed that the actual ages at which agencies were retiring buses from service exceeded FTA's minimum service life and suggested that the minimum service life policy needed to be changed (Laver, Schneck, Skorupski, & Cham, 2007).

Another 2007 report, "NCHRP 20-68: Domestic Scan Pilot Program Best Practices in Transportation Asset Management," identified the best-case practices and the asset management principles for transit agencies (Meyer & Cambridge Systematics, Inc., 2007). The Federal Highway Administration (FHWA), the American Association of State Highway and Transportation Officials (AASHTO), and the National Cooperative Highway Research Program

(NCHRP) sponsored this program. This research report was organized into four segments: a) case studies of some state transit agencies, b) case studies of local agencies and metropolitan planning organizations, c) observations of scan trips, and d) suggestions for further actions and research (Meyer & Cambridge Systematics, Inc., 2007).

Then, in 2008, the FTA published "Transit State of Good Repair: Beginning the Dialogue," the first step to collaborate transit asset management practices and provide strategies to address the state of good repair needs and transit asset management for the nation's transit rail and bus rolling stock (FTA, 2008). To do this, the FTA first convened a workshop in the summer of 2008, bringing together diverse stakeholders from 14 public transit providers and state departments of transportation to address the state of good repair for the nation's transit inventory. The objective of the workshop was to encourage stakeholders to be proactive by raising awareness regarding the scope of the problem and exploring creative approaches for funding of replacement and rehabilitation of aging transit assets. In the workshop, the FTA discussed the condition of transit capital assets, asset management practices, preventative maintenance practices, maintenance issues, and innovative financing strategies. The FTA also discussed related research work and supporting tools for transit agencies for coping with the state of good repair problems. The FTA further explained potential public-private partnership opportunities with manufacturers, engineering firms, and private equity firms for long-term capital asset management to make sure that the legacy assets are maintained and replaced when needed. While this workshop was successful in starting a useful dialogue among transit professionals, unfortunately the published report coming out of the round table workshop failed to define the state of good repair. The result of this omission is the FTA could not articulate how condition ratings, instead of age ratings, could be used effectively (FTA, 2008).

The "Rail Modernization Study," published in 2009 by the FTA, focused on capital expenditure and reinvestment needs for the nation's top seven transit agencies: Massachusetts Bay Transportation Authority (MBTA), Chicago Transit Authority (CTA), New Jersey Transit Corporation (NJ Transit), Metropolitan Transportation Authority of New York (MTA), Washington Metropolitan Area Transit Authority (WMATA), Southern Pennsylvania Transportation Authority (SEPTA), and San Francisco Bay Area Rapid Transit (BART) (Welbes, 2009). The study examined the asset management practices of the seven agencies and found that a backlog of $50 billion in 2008 would be needed to bring the seven agencies to the state of good repair, and an additional $5.9 billion would be required per year to maintain the state of good repair after that time. The FTA found that even though these agencies maintained their comprehensive asset inventories for capital funding, they lacked other asset management practices; relatively few transit agencies developed complete capital planning asset inventories to support long-term capital planning. Furthermore, the study found that, while only some of the largest transit agencies were making progress to improve their asset inventories, the relatively small and medium agencies had already developed these inventories. The shortcoming of this study was that the model did not consider future capacity expansion and other transit agency improvements (Welbes, 2009).

The 2010 "National State of Good Repair Assessment" study by the FTA was an expansion of the original 2009 "Rail Modernization Study" and evaluated the level of investment required to bring all agencies in the United States to a state of good repair (FTA, 2010a). This study showed that in 2009 an estimated SGR backlog of $77.7 billion would be needed to achieve the state of good repair and an additional $14.4 billion per year would be needed to maintain the normal replacement investment for a state of good repair. The study assessed the

national reinvestment needs considering the condition of the existing transit assets. The study found that about one-third of the nation's overall transit assets were either in marginal or poor condition, which meant these assets were either near or already exceeding their expected useful lives. However, when just bus and rail data were analyzed, 41% of bus assets and 26% of rail assets were either in marginal or poor condition. The report also described the methods for estimating the amount of investment, data sources, useful life assumptions, and type of investments required for the state of good repair needs. Furthermore, the study team also documented the processes, methods, and asset management practices of the study's 23 transit agencies that provided capital planning asset inventory data for long-term capital planning in support of both "National State of Good Repair Assessment" study and the earlier "Rail Modernization Study" (FTA, 2010a).

The 2011 synthesis, "TCRP Synthesis 92: Transit Asset Condition Reporting – A Synthesis for Transit Practice," documented the current transit asset management system practices for transit agencies as well as the local, state, and federal funding partners (McCollom & Berrang, 2011). This synthesis showed that large transit agencies use elementary asset management systems to fight against the consequences of underinvestment. Even though most large transit agencies had asset management systems that recorded all their assets, their systems were not able to make predictions about asset replacement under various funding scenarios. Finally, this synthesis provided several suggestions for improving the design and structure of the database, analysis techniques, and the SGR based tools for prioritizing funds (McCollom & Berrang, 2011).

The "Transit Cooperative Research Program (TCRP) Report 157: State of Good Repair - Prioritizing the Rehabilitation and Replacement of Existing Capital Assets and Evaluating the

Implications for Transit," report published in 2012 provided an SGR framework to evaluate and prioritize the rehabilitation and replacement investment decision for transit assets (Cohen & Barr, 2012). This SGR framework helps decision makers to answer questions regarding transit asset replacement and rehabilitation investment decisions. The report supported the framework by presenting an analytical approach along with a set of spreadsheet tools. The tools are intended for evaluating rehabilitation and replacement investments in specific transit assets and for prioritizing them. In conclusion, transit agencies will find these models a valuable resource to plan or finance public transportation (Cohen & Barr, 2012).

The "Moving Ahead for Progress in the 21st Century (MAP-21)," law was passed in July 6, 2012 and authorized $10.6 billion in the fiscal year 2013 and $10.7 billion in the fiscal year 2014 for federally funded transit agencies and highway programs (US Congress, 2012). Under the MAP-21 law, most of the funding was distributed through the core formula programs. MAP-21 created a state of good repair program and authorized at $2.1 billion in the fiscal year 2013 and $2.2 billion in the fiscal year 2014 for this program. Furthermore, the program also established new asset management systems and performance measurements for transit agencies (US Congress, 2012).

Thus, by 2014 the industry had published several research studies on state of good repair. Several studies were successful in addressing the problems of the state of good repair; however, more studies still needed to be completed. So, then in 2014 the "TCRP Project E-09: Guidance for Applying the State of Good Repair Prioritization Framework and Tools," provided guidance on how the framework and tools from 2012 TCRP Report 157 could be applied to evaluate and prioritize investment decisions in order to achieve a state of good repair. This research report

improved the framework and tools and then demonstrated their applications through a set of pilot programs and workshops (Robert, William; Reeder, Virginia; Lauren, Katherine, 2014).

A concurrent report, the 2014 "TCRP Report 172: Guidance for Developing a Transit Asset Management Plan," provided a system of how a transit asset management plan (TAMP) could be developed for use by transit agencies to achieve a state of good repair in accordance with the requirements of MAP-21 (Robert, Reeder, Lawren, Cohen, & O'Neil, 2014). This research was an expansion of the 2012 TCRP Report 157 and was intended to develop tools for transit agencies for the state of good repair. This TCRP 172 research introduced transit asset prioritization tool (TAPT), which consists of four spreadsheets tools for all types of transit assets (Robert, Reeder, Lawren, Cohen, & O'Neil, 2014). Although these tools were not as successful as expected, nevertheless these TAPT models were available to transit agencies to use for forecasting the future condition of transit assets and prioritizing rehabilitation and replacement investments.

This ten years of reports, research studies, round tables, workshops, and the MAP-21 provision culminated in the 2015 "Fixing America's Surface Transportation Act" or the FAST Act. This law reauthorized the public transportation and federal highway programs for the fiscal years 2016 to 2020 (APTA, 2016). The SGR saw a 23.9% increase by 2020 fiscal year beginning at $2.507 billion in 2016 fiscal year and rose to $2.684 billion by 2020 fiscal year. However, the FAST Act did not make significant changes in the SGR program to maintain the state of good repair on public transportation systems. In another case, the FAST Act incorporated about 2.85% of the total program funds for a High-Intensity Motorbus Vehicle State of Good Repair program. The FAST Act also suggested a maximum of 80% federal share for this program (APTA, 2016).

This dissertation focuses on the following aspects of the state of good repair development timeline. The early research from 2005 to 2006 on the state of good repair stated analytical tools for asset management, performance measures for transit asset management, and best practices to set the performance target. In 2007, the early research assessed the policy on existing minimum service life for transit buses and vans, and it suggested the minimum service life policy needed to be changed. Another report provided asset management principles for transit agencies. In 2008, the roundtable report provided strategies to address the state of good repair and transit asset management. In 2009, the study examined transit asset management practices of the seven agencies, and their capital expenditure and reinvestment needs. In 2010, the study assessed the reinvestment needs and evaluated the level of investment to bring all agencies to state of good repair. In 2011, the synthesis reviewed the current transit asset management system practices and provided suggestions to improve them. In 2012, the report provided a framework to prioritize the rehabilitation and replacement investment needs for transit assets. In 2014, the research reports provided guidelines to evaluate and prioritize investment decisions and provided a system to develop a transit asset management plan to achieve state of good repair. And finally, in 2015, FAST Act law authorized $2.507 billion in 2016 fiscal year and rose to $2.684 billion by 2020 fiscal year for the state of good repair program.

2.2. Overview of Transit Asset Management

According to section 1103 of MAP-21, asset management is defined as a set of "actions that will achieve and sustain a desired state of good repair over the lifecycle of the assets at minimum practicable cost" (Cevallos, 2016, p. 3). The FTA defines transit asset management as "Transportation asset management as a strategic and systematic process through which an organization procures, operates, maintains, rehabilitates, and replaces transit assets to manage

their performance, risks, costs over their lifecycle to provide cost-effective, reliable and safe service to current and future customers" (Lauren & Rose, 2012, p. 10). The FTA definition shows that asset management not only manages cost, it also handles risk and the performance across the lifecycle of transit assets (Lauren & Rose, 2012). Figure 1 shows how the ongoing asset management processes are related to cost, risk and system performance over the lifecycle of assets. The objective of asset management is to minimize the total cost as well as maximizing the performance (Rose, Lauren, Shah, Blake, & Parsons Brinckerhoff, 2012).

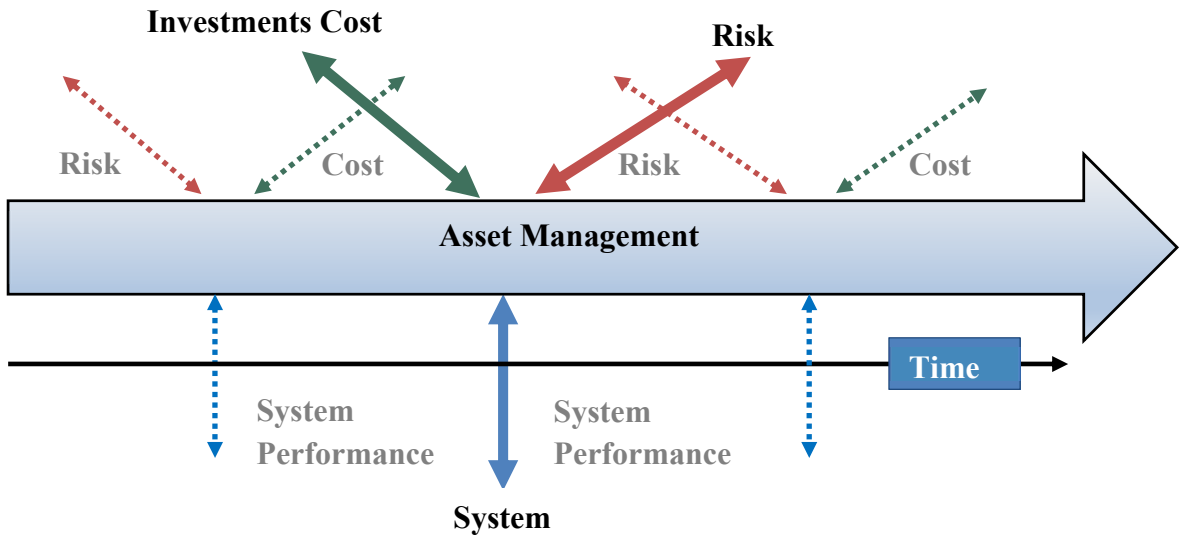


Figure 1. Asset Management Processes (Adapted from Rose, David, Isaac Lauren, Keyur Shah, Tagan Blake, and Inc. Parsons Brinckerhoff. 2012. *Asset Management Guide: Focusing on the Management of Our Transit Investments.* FTA Report No. 0027, U.S. Department of Transportation, Washington, D.C.: Federal Transit Administration. https://www.transit.dot.gov/about/research)

The FTA provides financial assistance to transit agencies to maintain their transit infrastructure and assets in a state of good repair. But the task is not easy for transit agencies because of the costs involved in other transportation assets such as bridges, highways, and transportation facilities. This is the reason transit agencies put more emphasis on asset management systems to manage their assets accurately (FTA, 2010b). Thus, MAP-21 requires

transit agencies to establish a transit asset management system. The development of an asset management system helps transit agencies to request needed funds for investments and attain a state of good repair (Cevallos, 2016). In addition, asset management systems can help transit agencies monitor their current assets' conditions and redistribute their existing resources to more effective uses (Meyer & Cambridge Systematics, Inc., 2007). Again, asset management can help agencies to prioritize capital investment, allocate limited resources to maintain current transit assets, and plan for replacement and rehabilitation of existing assets. In addition, asset management can help transit agencies optimize limited funding, estimate a state of good repair backlog, and set spending priorities (US GAO, 2013).

2.2.1. Key components of transit agencies' strategic management processes

Transit agencies need to manage their transit assets on a regular basis. Therefore, along with performance management and risk management, asset management has become an essential part of an agency's strategic management to achieve effective, high-level performance. Figure 2 shows the interaction among the agency's strategic management and its components. Transit agencies can accomplish their goals and objectives by combining and practicing these management processes. Individually, these management processes cannot be effective; they must be used in conjunction with the other management processes (APTA, 2013b; Cevallos, 2016).

Figure 2. Components of an Agency's Strategic Management Processes (Adapted from APTA. 2013b. *Creating a Transit Asset Management Program: Recommended Practice.* APTA-SGR-TAM-RP-001-13, Washington, DC: American Public Transportation Association, Working Group: Transit Asset Management)

2.2.2. Transit asset inventory development

As per the federal requirement for funding, transit agencies need to focus on the data-driven approach to measure the state of good repair and they need to require a transit asset inventory as the primary source of data (Cevallos, 2016). The asset inventory should include detailed information on the agency's assets and the assets' key attributes, such as asset type, asset age, expected useful life, and lifecycle costs. Figure 3 shows key steps of asset inventory development. The first step is to establish the organizational high-level class hierarchy for transit agencies to develop an asset inventory. The second step is to determine the asset inventory fields based on data requirements. The third step is to collect data, making sure that the data collection is consistent and accurate. After obtaining the necessary data, transit agencies must set the useful life and cost factors. The fourth step is to perform the quality check to ensure data accuracy. The

final step is to implement continuous improvement for data maintenance and constant evaluation (Cevallos, 2016).

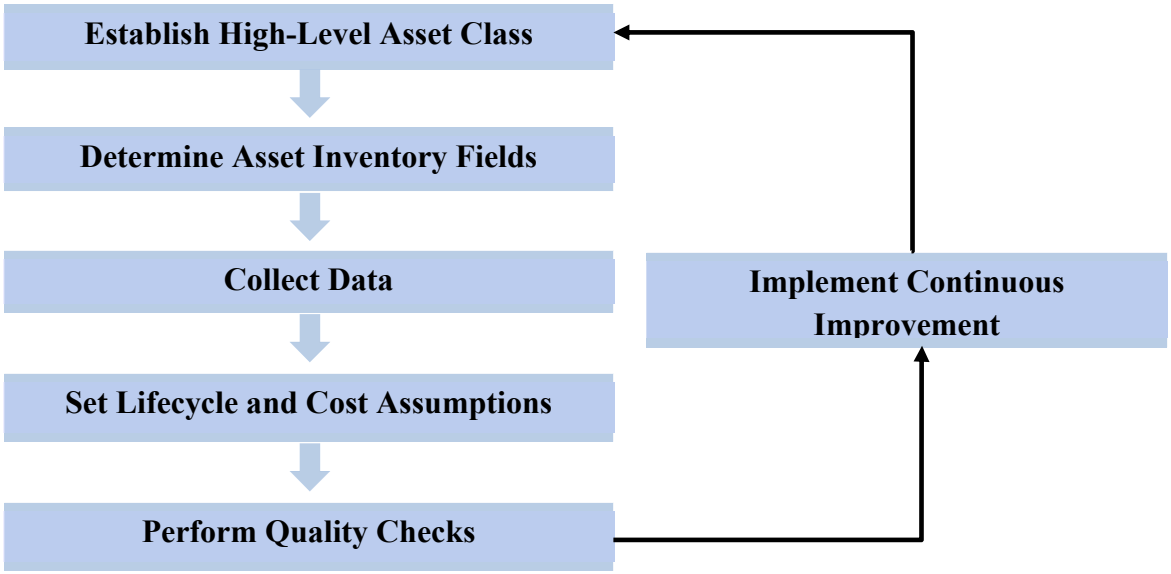


Figure 3. Asset Inventory Development Key Steps (Adapted from APTA. 2013b. *Creating a Transit Asset Management Program: Recommended Practice.* APTA-SGR-TAM-RP-001-13, Washington, DC: American Public Transportation Association, Working Group: Transit Asset Management)

2.2.3. Service life of transit asset

The FTA established a minimum useful life policy for transit vehicles funded with federal grants (Laver, Schneck, Skorupski, & Cham, 2007). The policy is to ensure that federally funded vehicles have a significant service life serving transit riders. The service life starts when the vehicle begins service and ends when it finishes service. The FTA's minimum service life varies by vehicle categories. Table 2 provides the vehicle categories and their minimum service life schedules. The service life of vehicles within different categories differs significantly. The 12-year bus category accounts for more than 25% of the nation's transit vehicles, while 4-year vehicle category accounts for 20% of the nation's transit vehicles. The analysis on 12-year category vehicles shows that the average age is 15.1 years which means most transit agencies operate buses above the minimum service life (Laver, Schneck, Skorupski, & Cham, 2007).

Table 2. Transit Vehicle Minimum Service Life

Category	Typical Characteristics				Minimum Life Whichever comes first	
	Length	Approximate Gross Vehicle Weight	Seats	Average Cost	Years	Miles
Heavy Duty Large Bus	35 - 48 feet and 60 feet Articulated	33,000 to 40,000	27 to 40	$325,000 to over $600,000	12	500,000
Heavy Duty Small Bus	30 feet	26,000 to 33,000	26 to 35	$200,000 to $325,000	10	350,000
Medium Duty and Purpose-Built Bus	30 feet	16,000 to 26,000	22 to 30	$75,000 to $175,000	7	200,000
Light Duty Mid-Sized Bus	25 to 35 feet	10,000 to 16,000	16 to 25	$50,000 to $65,000	5	150,000
Light Duty Small Bus, Cutaways, and Modified Van	16 to 28 feet	6,000 to 14,000	10 to 22	$30,000 to $40,000	4	100,000

Source: Adapted from Laver, Richard, Donald Schneck, Douglas Skorupski, and Laura Cham. 2007. *Useful life of transit buses and vans.* No. FTA-VA-26-7229-07.1, U.S. Department of Transportation, Washington, D.C.: Federal Transit Administration.

Based on an analysis of the average retirement age of transit assets on NTD data, the FTA found that the average retirement age was longer than the minimum required age in practice (Edrington, et al., 2014). The NTD database contains the statistics of national transit vehicles. Table 3 provides the average vehicle retirement age by vehicle category. The average retirement age of 4-year van is 5.6 years with 29% of the vehicle retired one or more years after the FTA minimum retirement age. Table 3 also shows that about 20% of 5 and 12-year vehicles exceed one or more year past the minimum retirement age. Besides, 10% of 4-year vehicles exceed three or four years past the minimum retirement age (Edrington, et al., 2014).

Table 3. Actual Average Vehicle Retirement Age

Vehicle Category with Minimum Retirement Age	Average Retirement Age (Years)	Share of Active Vehicles That Are:	
		One or more years past the minimum service life	Three or more years past the minimum service life
12 – Year Bus	15.1	19%	9%
10 – Year Bus	8.4	7%	4%
7 – Year Bus	8.2	12%	3%
5 – Year Bus/Van	5.9	23%	5%
4 – Year Van	5.6	29%	10%

Source: Table adapted from Laver, Richard, Donald Schneck, Douglas Skorupski, and Laura Cham. 2007. *Useful life of transit buses and vans.* No. FTA-VA-26-7229-07.1, U.S. Department of Transportation, Washington, D.C.: Federal Transit Administration.

2.3. Condition of the United States Transportation System

There are about 850 urban transit agencies, and 1700 rural and tribal transit agencies provide transportation services by transit bus, commuter rail, light rail, ferryboat, and subway. Table 4 below shows that public transit provided about 10.5 billion unlinked trips in 2014 which is an increase of 20.5% over 2000 (BTS, 2016).

Table 4. Transit Vehicles and Ridership: Unlinked Passenger Trips

Fiscal Year	2000	2010	2013	2014
Unlinked Passenger Trips (Billions)				
Heavy Rail	2.63	3.55	3.82	3.93
Commuter Rail	0.41	0.46	0.48	0.49
Light Rail	0.32	0.46	0.52	0.48
TOTAL, Rail Transit UPT	3.36	4.47	4.81	4.90
Motor Bus	5.16	5.24	5.33	5.04
Demand Response	0.07	0.10	0.11	0.10
Ferry Boat	0.05	0.06	0.06	0.06
Other	0.08	0.10	0.09	0.40
TOTAL, Non-Rail Transit UPT	5.36	5.49	5.60	5.61
TOTAL, Transit UPT	8.72	9.96	10.41	10.51

Source: Table adapted from BTS. 2016. *Transportation Statistics Annual Report.* U.S. Department of Transportation, Washington, D.C.: Bureau of Transportation Statistics.

Table 5 shows the average age of vehicles from 2000 to 2014 (BTS, 2016). The average age of commuter rail passenger coaches increased over that period. The average age of the heavy-rail passenger car fleet was 20.4 years old in 2014 but decreased by 2.5 years between 2000 and 2014. The average age of the transit buses was 7 to 8 years, and the average age of light-rail vehicles was near 17 years. The bus fleet stayed comparatively newer than the transit rail fleet as many transit agencies either retired, replaced or added new vehicles to the fleet and the rail cars lasted longer than buses (BTS, 2016).

Table 5. Transit Vehicles and Ridership: Average Age of Vehicles

Fiscal Year	2000	2010	2013	2014
Average Age of Vehicles				
Heavy Rail Passenger Cars	22.9	18.7	20.2	20.4
Commuter Rail Passenger Coaches	16.9	18.9	20.8	18.8
Full Size Transit Buses	8.1	7.9	8.1	7.2
Light Rail Vehicles	16.1	16.8	16.4	16.7
Transit Vans	3.1	3.4	3.5	3.5
Ferry Boats	25.6	20.5	21.4	23.8

Source: Table adapted from BTS. 2016. *Transportation Statistics Annual Report.* U.S. Department of Transportation, Washington, D.C.: Bureau of Transportation Statistics.

In 2014, transit riders made about 10.5 billion trips which were 5.5% increases from 2010 (BTS, 2016). Table 6 shows that the transit riders traveled about 57.0 billion miles in 2014 which were 8.2% travel increases since 2010. The light rail, commuter rail, and heavy rail made up the nation's rail transit with 15.3% of the total transit vehicles. The rail transit made 46.6% of the total trips, and 57.2% of the total person-miles traveled. The bus transit produced 47.9% of total transit trips and 37.9% of the total person-miles (BTS, 2016).

Table 6. Transit Vehicles and Ridership: Person-Miles Travelled

Fiscal Year	2000	2010	2013	2014
	Number of Transit Vehicles			
Heavy Rail Cars	10,311	11,510	10,380	10,551
Commuter Rail Cars and Locomotives	5,497	6,768	7,150	7,177
Light Rail Cars	1,306	2,096	2,842	2,444
TOTAL, Rail Transit Vehicles	17,114	20,374	20,372	20,173
Motor Bus	59,230	63,679	66,823	62,449
Demand Response	22,087	33,555	31,433	31,359
Ferry Boat	98	134	156	144
Other	7,607	18,066	17,793	17,850
TOTAL, Non-Rail Transit Vehicles	89,022	115,434	116,205	111,802
TOTAL, Transit Vehicles	106,136	135,808	136,577	131,974
	Person Miles (Millions)			
Heavy Rail	13,844	16,407	18,005	18,339
Commuter Rail	9,400	10,774	11,736	11,600
Light Rail	1,339	2,173	2,565	2,675
TOTAL, Rail Transit PMT	24,583	29,353	32,305	32,614
Motor Bus	18,999	20,739	21,414	21,587
Demand Response	588	874	852	864
Ferry Boat	298	389	402	414
Other	632	1,315	1,449	1,534
TOTAL, Non-Rail Transit PMT	20,517	23,317	24,117	24,399
TOTAL, Transit PMT	45,100	52,670	56,422	57,013

Source: Adapted from BTS. 2016. *Transportation Statistics Annual Report.* U.S. Department of Transportation, Washington, D.C.: Bureau of Transportation Statistics.

2.4. Analytical Tools for State of Good Repair

The Map-21 authorized, and the FAST Act reauthorized FTA to develop a rule for the state of good repair program. This rule establishes a system to monitor performance, manage transit assets, increase safety and reliability, and estimate performance measures (WSDOT, 2016). Therefore, transit agencies need to develop a TAMP process per MAP-21 and FAST Act requirements to achieve a state of good repair. FTA also developed TAPT tool for transit agencies to support the TAMP process. This TAPT tool includes four spreadsheet models which help transit agencies to predict the future conditions of their transit assets and help prioritize rehabilitation and replacement needs. The FTA's TERM Lite can be used along with TAPT or

without TAPT to support analysis of different investment scenarios. Furthermore, many agencies developed their decision support tools and an asset management system which can be used to support TAMP processes (Robert, William; Reeder, Virginia; Lauren, Katherine, 2014).

2.4.1. FTA's transit economic requirements model (TERM Lite)

The FTA developed Transit Economic Requirements Model (TERM Lite) tool in 1995 to estimate transit capital needs and spent about $5 million in development and update until 2013. The TERM model measures asset condition on a 5-point scale and considers a revenue vehicle to be in a state of good repair if the condition of the vehicle reaches or above the condition rating of 2.5 (FTA, 2013; Zarembski, 2013). It estimates the state of good repair backlog, determines the capital funding levels required to achieve the state of good repair, analyze the impact of projected future investment on capital performance, and prioritize long-term investment (Cevallos, 2016). By using TERM, the transit agencies can forecast the trend of asset maintenance, replacement, and rehabilitation costs for the next 20-year period as well as the FTA can estimate the capital needs and develop various reports. The TERM model uses information obtained from NTD database. The asset age and physical condition for each asset category are considered as the predictor for determining the condition (Cevallos, 2016).

The TERM model can predict a current and future asset condition based on a five-point rating system as shown in Table 7 (FTA, 2010a). It uses the numerical method to rate transit asset condition based on a scale of 5.0 for excellent, 4.0 for good, 3.0 for adequate, 2.0 for marginal, and 1.0 for poor for evaluating a transit asset condition based on their age, replacement or rehabilitation history, and other factors. If the rating of the asset is at or above the condition rating of 2.5, TERM model considers it a state of good repair. Similarly, if the condition value of

all transit assets is 2.5 or higher in a transit agency, it will be considered in a state of good repair (FTA, 2010a).

Table 7. TERM Condition Ratings

Condition	Ratings	Description
Excellent	5.0 to 4.8	New or like new asset
Good	4.7 to 4.0	Asset showing minimal signs of wear
Adequate	3.9 to 3.0	Asset has reached mid-life
Marginal	2.9 to 2.0	Asset reaching or just past its useful life
Poor	1.9 to 1.0	Asset past its useful life

Source: Adapted from FTA. 2010. *National State of Good Repair Assessment.* Report to Congress, U.S. Department of Transportation, Washington, D.C.: Federal Transit Administration.

2.4.2. Other analytical tools for state of good repair

Along with the TERM tool, the FTA also developed four analytical tools for transit agencies to support the TAMP process. These tools are (1) prioritization modeling tool, (2) vehicle modeling tool, (3) age-based modeling tool, and (4) condition-based modeling tool. They are described below.

1. **Prioritization Modeling Tool**

This tool prioritizes a series of asset rehabilitation or replacement funds and simulates the funds for ten years (Cohen & Barr, 2012). This tool provides a set of recommendations for the investment plan based on the allocated budget and prioritization index (PI) results. Even though the tool provides the straightforward approach for allocating funds for replacement and rehabilitation based on PI, in practice higher-ranked projects with available budgets may need to be rescoped, and smaller projects need to be combined. Also, there might be a limitation of maximum and minimum spending by asset category to get a reliable solution (Cohen & Barr, 2012).

2. Vehicle Modeling Tool

The vehicle modeling tool estimates the cost minimizing point that a bus or rail vehicle should be replaced and predicts the annual costs and prioritizes replacement of transit vehicles based on age (Cohen & Barr, 2012). It considers energy or fuel costs, rehabilitation costs, and delay costs for calculating the need for replacement or rehabilitation. Transit agencies should use this tool multiple times as the calculations are fleet specific. Therefore, the transit agencies need to develop various models for different vehicle types (Cohen & Barr, 2012).

3. Age-Based Modeling Tool

The age-based modeling tool assesses deteriorations on transit asset other than a transit vehicle over time and forecasts the annual costs of the transit agency as well as user costs of the transit asset (Cohen & Barr, 2012). It also prioritizes asset replacement based on a function of age. This tool calculates asset replacement cost and predicts when the asset will fail if it is not replaced. It is intended to use for different asset types other than vehicles. Therefore, the transit agencies should use this tool multiple times for different asset types. The age-based model may not be preferable in some complicated situation where age might be a poor predictor of an asset. However, the age-based model requires comparatively less data than the other models (Cohen & Barr, 2012).

4. Condition-Based Modeling Tool

The condition-based modeling tool uses on non-vehicle assets that deteriorate as a function of condition (Cohen & Barr, 2012). It predicts the annualized user costs of the assets to the transit agency. Using this tool, the rehabilitation or replacement actions are performed on the transit asset based on priority, and condition. This tool is intended to use for specific multiple non-vehicle assets, therefore transit agencies need to run it multiple times for multiple asset

types. Guideway, facilities, systems, and stations are modeled using the tool. The condition-based model is preferable in a complex situation where the condition is a good predictor rather its age (Cohen & Barr, 2012).

2.5. Review of Transit State of Good Repair Practices in the United States

Most of the transit agencies use TERM Lite as their leading practices for a state of good repair. They also use TERM Lite to collect data and develop information inventories to manage transit assets and prioritize capital investment. However, some of the transit agencies are using in-house assessment tools to estimate a state of good repair needs, make capital investment decisions on the state of good repair backlogs, prioritize rehabilitation and replacement needs (US GAO, 2013). A review of transit state of good repair practices and asset management practices in selected transit agencies are summarized below:

2.5.1. MARTA state of good repair

The Metropolitan Atlanta Rapid Transit Authority (MARTA) provides rail rapid transit and bus service to Atlanta area. In the 1990's, MARTA developed an integrated maintenance management information system (MMIS) which has a standalone asset database to track its assets but its limitation in functionality led to poor quality asset data. The asset condition reports are stored in the database which is collected through testing of preventive maintenance and field inspection. MARTA analyzes the data to determine its rehabilitation and replacement needs (Cohen & Barr, 2012). In 2006, it obtained an enterprise asset management(EAM) system and utilized the life cycle asset rehabilitation enhancement (LCARE) system to establish and improve its asset management system. In 2010, efforts were made to complete information on assets on an existing database and added missing assets in the database. However, the budget cuts increased the MARTA's SGR backlog (Springstead, 2011).

2.5.2. MBTA state of good repair

The Massachusetts Bay Transportation Authority (MBTA) developed an SGR database that includes its asset inventory and an application for predicting future asset replacement needs. The MBTA uses the database to prioritize the rehabilitation needs based on the age of the transit asset representing as the percent of useful life, operation impact, and cost-effectiveness. They also use the SGR database to describe the scale and scope of the state of good repair and backlog. MBTA prepares annual capital investment program (CIP) which includes a 5-year capital investment plan to maintain a state of good repair (Cohen & Barr, 2012; Waaramaa, 2010).

2.5.3. MTC state of good repair

The Metropolitan Transportation Commission (MTC) developed a comprehensive regional transit capital investment (RTCI) database for the Bay Area Transit. This database tracks all the transit assets on different transit agencies in the Bay Area. The database is also used to allocate the limited funding to the agencies in a consistent manner to replace the assets and make sure that the assets maintain its state of good repair. The RTCI built a classification on assets and included analysis tool for replacement needs. The tool provides the average lifespan for each asset category for replacing the assets. The RTCI provides the projection of the 25 years' transportation funding plan among nine counties in the Bay Area. The funding for each transit agency depends on the average age of assets as a percentage of their useful life (AAAPUL), a measurement of asset conditions and objectives to reach a state of good repair (Cohen & Barr, 2012).

2.5.4. NJDOT state of good repair

The New Jersey Department of Transportation (NJDOT) coordinates with New Jersey Transit (NJT), South New Jersey Transportation Authority (SJTA) and New Jersey Turnpike Authority (NJTA) and produces its capital investment strategy (CIS). The CIS allocates the transportation funding for the next ten years for transit assets. NJDOT categorizes its total assets into nine classes and assigns a set of goals in each category. The CIS monitors how the system performance varies over time with different funding scenarios and performs trade-off analysis with different investment strategies (Cohen & Barr, 2012). The CIS developed an asset management decision support model. The model assists NJDOT to use asset data and systems to make high-level resource allocation decisions. It also helps to use available data to prioritize problems (Louch, Robert, Gurenich, & Hoffman, 2009).

2.5.5. RTA state of good repair

The Regional Transportation Authority (RTA) supervises all public transportation in Northern Illinois. RTA also provides planning and allocates funding to Pace Suburban Bus, Chicago Transit Authority (CTA), and Metra Commuter Rail. RTA's asset management system has the SGR needs-assessment process which is based on an ongoing inventory condition assessment program. The system contains a capital plan development process which links to ongoing performance measurement so that the authority can analyze and prioritize investment funding. In addition to this program, RTA includes an integrated decision support tool along with the FTA's TERM model (FTA, 2011).

2.5.6. CalTrain state of good repair

The Peninsula Corridor Joint Powers Board (JPB) operates the CalTrain which is a commuter railroad servicing the community from San Francisco to Gilroy since 1992. The JPB

developed the CalTrain asset management system (CTAMS) to bring CalTrain rails into a state of good repair. The CTAMS tracks the condition of transit assets and keeps maintenance records. It also helps to make a decision on prioritizing and coordinating replacement and rehabilitation needs within existing budgets. It considers factors such as the age of transit assets, standard requirements of Federal Railroad Administration (FRA), and the SGR standard requirement of CalTrain to measure transit asset conditions (FTA, 2011).

2.5.7. NYCT state of good repair

The New York City Transit (NYCT) initiated its SGR program by developing a database which tracks its asset and prioritize its capital investment needs. A detail information about an asset in input in the database which enables NYCT to identify the specific assets which require capital investment. This information helped NYCT to plan 5 years for capital investment and 20 years for needs planning and acquire significant progress in restoring the agency's assets to a state of good repair. NYCT also initiated a new condition-based approach to replace or rehabilitate their transit assets. In this approach, they determine the asset condition based on the asset's condition ranking its age versus the remaining usage life, and the actual asset performance (McCollom & Berrang, 2011). In its capital investment program, it allocates SGR reinvestment needs to correct past maintenance or replace equipment which have no useful life (FTA, 2010b).

2.5.8. VDOT state of good repair

The Virginia Transportation system consists of Virginia Department of Transportation (VDOT), Department of Rail and Public Transit (DRPT), Department of Aviation (DOAV), and Virginia Port Authority (VPA). VDOT and DRPT both have developed a performance dashboard while VDOT measures asset condition, DRPT measures its ridership. DRPT developed a transit

asset management system (Trans-AM) which helps the FTA to facilitate the state of good repair practices throughout the transit agency. Virginia took a leadership role in transit asset management with the recent development of program guidance and grant evaluation system (PROGGRES). DRPT implemented PROGGRES that effectively address the capital needs and policy for the state of good repair programs (Cambridge Systematics, 2009). To support and in accordance with fulfilling state and federal requirements as asset management, VDOT established a detailed asset management method which measures the performance and manages transit assets based on life cycle approaches and allocate funds to different transit agencies based on needs-based budget approach (VDOT, 2006).

2.5.9. WMATA state of good repair

The years of underfunding and the tremendous regional growth caused underinvestment in Washington Metrorail's Area Transit Assets and created unreliable services for riders. Therefore, WMATA created momentum which is a strategic 10-year capital investment plan to bring their transit assets into a state of good repair. Momentum planned Metro 2025 with $6 billion of critical capital investment to maximize the existing rail system, improve the rail stations and pedestrian connection, enhance bus service, upgrade communication systems, expand maintenance facilities, and improve the transit infrastructure. With the first capital investment, WMATA estimates a capacity increase of 36000 passengers per hour during rush hour. With its second investment which is a "quick win," WMATA relieves crowding in its largest bottlenecks and brings the system to a state of good repair (MTA, 2014).

2.6. Summary of Best Practices of SGR on Selected Transit Agencies

Here are some highlights of best practices from each agency in the Table 8:

Table 8. SGR Best Practices

Agency	Business Process	Best Practices
CTA (Chicago)	• Strategy • Condition Assessment • Performance Monitoring • Capital Programming	• Set up performance measures for each category • Evaluate asset condition consistently across all assets • Align condition & performance • Apply formal process for capital projects for asset life-cycle
MARTA (Atlanta)	• Inventory • Lifecycle Management • Capital Programming • Predictive modeling	• Develop formal asset management plans • Apply capital programming process considering asset conditions, remaining service life, and lifecycle costs • Evaluate state of good repair analysis and the SGR backlog
MBTA (Massachusetts)	• Capital Programming • SGR Database	• Establish the annual CIP for 5-year investment plan • Build the SGR database which estimated SGR backlog, and prioritize the rehabilitation needs
MTC (Bay Area)	• Capital Programming • Lifecycle Management	• The RTCI database tracks transit assets and allocate the funding to the transit agencies. • The tool provides the average lifespan for each asset category, projects costs for replacing the assets.
NJDOT (New Jersey)	• Strategy • Asset Management	• The CIS allocates the transportation funding for the next ten years for transit assets. • The NJDOT categorizes its total assets into nine classes and assigns a set of goals in each category. • The asset management decision support model assists NJDOT to use asset data and systems to make high-level resource allocation decisions and prioritize problems.
CalTrain (San Francisco)	• Inventory • Asset Management	• The CTAMS use Microsoft Excel to track the condition of transit assets, helps to decide on prioritizing and coordinating replacement and rehabilitation needs within existing budgets.

Table 8. SGR Best Practices (continued)

Agency	Business Process	Best Practices
NYCT (New York City)	• Asset Management • Condition assessment • Capital investment programming	• The SGR database tracks its asset and prioritizes its capital investment needs. • The capital investment program allocates SGR reinvestment needs to correct past maintenance or replace equipment which has no useful life
VDOT (Virginia)	• Condition Measurement • Asset management • Policy • Lifecycle management	• Developed the PROGGRES which helps the FTA to facilitate the state of good repair practices throughout the transit industry. • PROGGRES address the capital needs and policy issues associated with the state of good repair programs. • VDOT established an asset management method which measures the performance and manages transit assets based on life cycle approaches and allocate funds to different transit agencies based on needs-based budget approach.
WMATA (Washington)	• Capital programming	• WMATA created momentum which is a strategic 10-year CIP plan to maximize the existing rail system, improve the rail stations and pedestrian connection, enhance bus service, upgrade communication systems, expand maintenance facilities, and improve the transit infrastructure.

Source: *Adapted from APTA. 2013.* Defining a Transit Asset Management Framework to Achieve a State of Good Repair: Recommended Practice. *APTA SGR-TAM-RP-002-13, Washington, D.C.: APTA Standards Development Program Working Group: Transit Asset Management; FTA. 2010b.* Transit Asset Management Practices: A National and International Review. *FTA Report, U.S. Department of Transportation, Washington, D.C.: Federal Transit Administration.*

2.7. Summary of Literature Review

The literature review conducted in this research found that the Federal Transit Administration was trying to find an intelligent way to solve the transit state of good repair that nation's transit agencies were facing. Per Map-21 requirements, transit agencies require a predictive model for prioritizing capital investment for replacement and rehabilitation of transit

vehicles. The FTA conducted case studies on several transit agencies and found that most of the transit agencies were lacking asset management practices and didn't have complete transit asset inventories. Another study by the FTA on the useful life of transit buses and vans showed that the minimum service life policy by the FTA might need to be changed. The NCHRP report indicated that two analytical tools could be used along with existing systems to make the investment decision on transit vehicles. Another report by NCHRP showed that how well a performance measure could be used for decision-making process for capital investment. The TCRP report 157 provided a framework for the state of good repair and developed tools for evaluating and prioritizing investment. The TCRP project E-09 improved the state of good repair framework which was developed in TCRP report 157. The TCRP project E-09 provided guidance on how the framework and tools can be used to achieve the state of good repair. As a continuation of TCRP report 157, the TCRP report 172 developed a transit asset management plan in accordance with the Map-21 requirements and further improved the prioritization tools for transit agencies. The TCRP synthesis 92 showed that most transit agencies were not able to make replacement decisions under different funding scenarios. Another NCHRP 20-68 pilot program provided best practices in transportation asset management for transit agencies.

The literature review also discussed on transit asset management system and how it helps transit agencies to maximize system performance. The transit asset management is very important for transit agencies as the Map-21 requires them to build transit asset management system to get the federal funds. Therefore, transit agencies need to develop transit asset inventory and asset management recommended several key steps to develop the asset inventory.
As per Map-21 and FAST Act, transit agencies are required to develop a transit asset management plan to achieve the state of good repair. Therefore, the FTA developed TAPT tools

as well as TERM tool to predict the condition of transit assets and prioritize the investment needs.

The literature review also reviewed the current state of good repair practices and asset management practices on nation's major transit agencies. The reviews showed that most of the agencies use their in-house analytical tool to estimate the state of good repair needs. However, most transit agencies do not have comprehensive transit inventories for asset management purposes. The MBTA uses their own SGR database and an application to predict future replacement and rehabilitation needs. The NYCT also uses their own SGR database to prioritize its capital investment needs. The WMATA uses 10-year capital investment plan to achieve the state of good repair. Finally, this chapter concludes by presenting several best practices for the state of good repair on nation's top transit agencies.

CHAPTER 3. METHODOLOGY

Based on the information from the SGR practices of the transit agencies as well as literature review, three predictive models were developed to address the state of good repair. In this research, the predictive models were developed by applying machine learning algorithms which predict the projected service life of transit vehicles and help decision-makers to evaluate replacement and rehabilitation needs for transit vehicles and allocate available funds across overall transit assets. Furthermore, this will also help to evaluate the long-term capital funding and its impact on the future condition as the performance of transit assets.

3.1. Basic Concept of Machine Learning Techniques

The field of machine learning builds a computer program which can automatically improve with experience (Jordan & Mitchell, 2015). It is one of the rapidly growing technologies, which uses the core concept of Artificial Intelligent (AI), data science, computer science, and statistics. The development of new machine learning algorithms and the availability of online data made the machine learning techniques more effective. Since machine learning methods are data intensive, the application of machine learning is an evidence-based decision-making process across science, technology, medical, education, manufacturing, financial, and marketing (Jordan & Mitchell, 2015).

Machine learning algorithms have been developed to solve data and machine learning related problems (Jordan & Mitchell, 2015). In the past decade, the scientists and engineers collected a vast amount of data through networking and mobile computing systems that are referred to as ‘big data.’ They used machine learning to convert these data for a solution to the problem. Machine learning algorithms learn from large amounts of data and customize the output based on business requirements. The trend of capturing and mining large amounts of diverse data

sets can improve services and productivity across many fields of science. For example, historical medical records can be used to identify a patient with similar symptoms and provide the best treatment; historical traffic data can be used to control traffic perfectly and reduce congestion; historical crime data can be used to allocate police to a specific location and reduce the crime rate. Therefore, many organizations are capturing large data sets and analyzing them through machine learning techniques to automate decision making processes across many aspects of data-intensive sciences (Jordan & Mitchell, 2015).

In general, there are three types of machine learning called supervised learning, unsupervised learning, and reinforcement learning (Raschka, 2015). In this methodology, the supervised learning would be utilized for the problem and described below.

3.1.1. Supervised learning

Supervised learning uses inductive methodologies and learns from input-output pairs (Shen & Chouchoulas, 2001). The supervised learning learns from labeled training data and makes the prediction to unseen data. Supervised learning is useful when systems under the training data are intended to perform as learning with real results. In this case, the results are known, but the rules to perform the tasks are not known. Therefore, the system needs to be trained by learning algorithms and examples, then apply the learning knowledge to the entire domain (Shen & Chouchoulas, 2001). One sub-category of supervised learning is the regression. In regression analysis, many predictor variables along with a continuous response variable are used to find a relationship between these variables to predict an outcome (Raschka, 2015). Figure 4 shows the workflow diagram of how supervised learning makes the prediction.

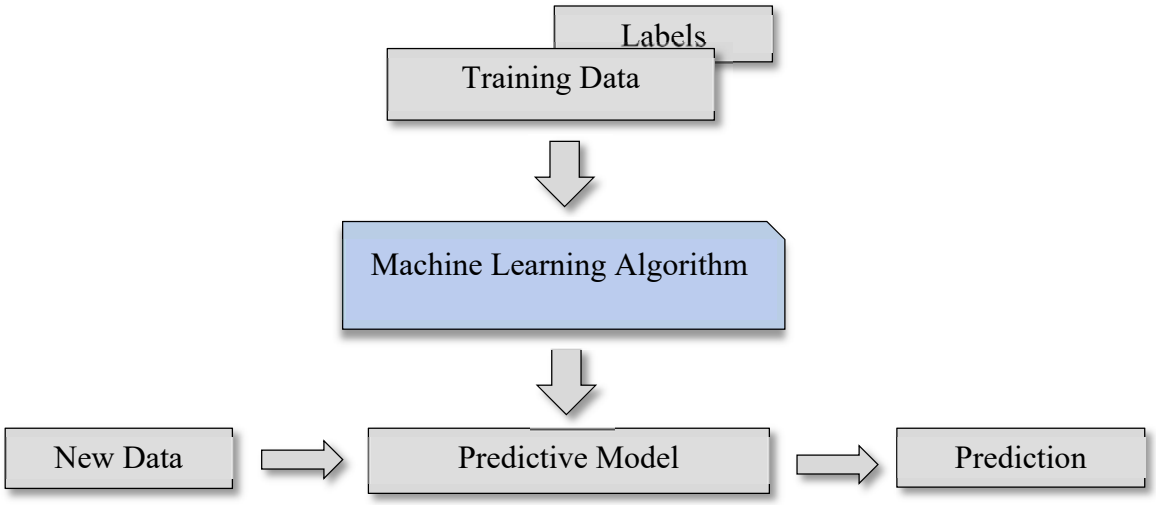


Figure 4. Making Prediction about the Future with Supervised Learning (Adapted from Raschka, Sebastian. 2015. *Python machine learning.* First Edition. Edited by Roshni Banerjee. Birmingham: Packt Publishing Ltd.)

3.2. Machine Learning Algorithms

In this research, three machine learning techniques have been used for estimating the service life of revenue vehicles and the best method has been selected to solve the state of good repair problem. For machine learning algorithms, a training set has been created with the revenue vehicle inventory data from the fiscal year 2008 to the fiscal year 2016 from NTD legacy database. The training data set has vehicles which had already been retired and stored training instances in the memory for prediction of the service life of non-retired vehicles and solve the state of good repair needs.

There are many methods of machine learning available for building predictive models. In this problem, the ensemble method had been used to build the SGR predictive model. In order to choose the best model for the problem, three kinds of comparative analysis of machine learning algorithms had been conducted. They are random forest regression, gradient boosting regression, and decision tree regression (Lee & Min, 2017). At first, the random forest regression had been applied, followed by gradient boosting regression, and finally, decision tree regression method.

After comparing the performance measurements amongst the model, the best predictive model had been chosen for the problem.

3.2.1. Ensemble methods

Ensemble methods are very powerful techniques, and the basic idea is to train multiple learners to solve the same problem and then combine them by averaging the output of models to calculate the final prediction. Therefore, ensemble methods are significantly more accurate than a single learner (Zhou, 2012). The idea of ensemble methods is used in many decision-making situations in our daily lives (Zhang & Haghani, 2015). For example, when we have problems, we seek others' opinions. By combining the weighted ideas, we can get a better decision. Therefore, the success of the ensemble method depends on the combination of base models. If individual base models generate different outputs, then combining several base models is useful. The ensemble methods minimize errors on the predictions by correcting mistakes on the predictions made by the individual base model. If individual base models produce similar mistakes, then combining base models is worthless. There are two techniques such as bagging and boosting which uses various resampling methods to achieve diverse base models (Zhang & Haghani, 2015).

Ensemble methods can handle extremely complicated behavior, but they are very simple to use and can rank features based on the predictive performance. Ensemble methods became successful in many real-world problems and provided nearly optimum performance among all major predictive analytics (Bowles, 2015; Zhou, 2012). The most popular ensemble algorithms are adaBoost, boosting, bootstrapped aggregation (Bagging), gradient boosting machines (GBM), stacked generalization (blending), gradient boosted regression trees (GBRT), and random forest (Brownlee, 2013).

Figure 5 outlines a common ensemble architecture. There are numerous learners in an ensemble which are called base learners. The training data generates base learners using the base learning algorithms such as neural networks, decision tree or other learning algorithms. In most of the cases, ensemble methods apply single base learning algorithm; however, some of the ensembles use multiple learning algorithms (Zhou, 2012).

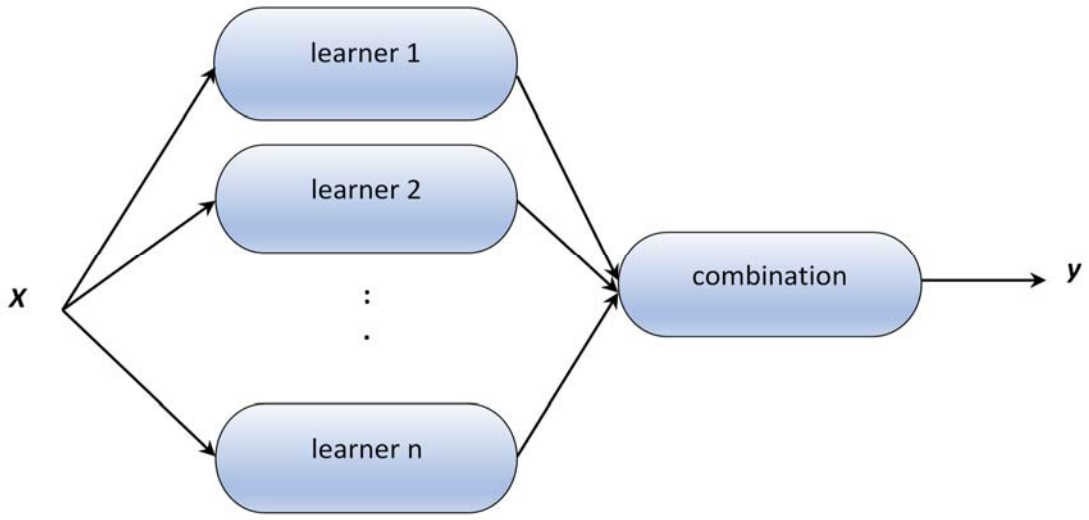


Figure 5. A Common Ensemble Architecture (Adapted from Zhou, Zhi-Hua. 2012. *Ensemble Methods: Foundations and Algorithms.* Edited by Ralf Herbrich and Thore Graepel. Boca Raton, FL: Chapman & Hall/CRC.)

The ensemble tree uses the averaging technique to reduce the variance. Both ensemble tree-based algorithms use a single regression tree as their base model. The random forest uses the bagging technique while the gradient boosting uses the boosting technique. In the boosting method, the base model appears sequentially, and the examples which are difficult to estimate in the previous base model appears in the training data more often than the ones which are correctly estimated. The additional base models will correct mistakes which were made in the previous base models. The gradient boosting regression method uses a forward stage-wise modeling approach which fits additional models to minimize the gap between the prediction value and the

true value by using the loss function such as squared error or an absolute error. In the regression problems, the boosting method uses a gradient descent optimization technique which minimizes specific loss function by adding a base model at each step to reduce the loss function accurately. The performance of the model can be optimized by the best combination of the parameters (Zhang & Haghani, 2015).

3.2.1.1. Random forest regression

Random forest is a predictive algorithm which is a representative of ensemble methods (Kumar, 2016). The algorithm creates predictions on individual trees randomly and then averages predictions of all trees. The random forest does not use the cross-validation process; instead, the method uses bagging. Suppose there are *m* number of variables, and *n* number of observations in training data set *T*. *S* number of trees need to be grown in the forest, and each tree will be grown from the separate training data set. Each training data set from S number of training data sets is created from sampling *n* observation randomly; therefore, some data sets might get duplicate observations, and some observations might be missing from all the *S* training data sets. These data sets are called bootstrap samples or bagging. The observations that are not part of the bag are "out of the bag" (Kumar, 2016). A random forest model has better generalization performance than an individual decision tree because of its randomness, and it helps the model to decrease the variance. Another advantage of random forest is that they are good at handling outliers in the data set and do not need much parameter optimization (Raschka, 2015).

3.2.1.2. Gradient boosting regression

Gradient boosting regression trees are stage-wise ensemble trees where weak models are fit sequentially to minimize the errors on the training set and predictions are made by the

previous model in the sequence (Gagne, McGovern, Haupt, & Williams, 2017). These weak models are considered as decision trees in gradient boosting trees. In the beginning, the initial model is fit directly to the training labels, and the additional weak models are fit sequentially to the negative gradient of the loss function to optimize the predictive model. The difference between the actual observation and the prediction from the previous model is called a residual, which is also the mean squared error of the loss function. The predicted residual is added to the sum of the previous residuals. A learning rate is multiplied by each tree's prediction to minimize the residual of the prediction, and a smaller learning rate can be used to correct the prediction and minimize the risk to fit to noise. The base gradient boosting regression model uses the default parameters of learning rate 0.1, 500 trees, a maximum depth of 5, and least absolute deviance loss function (Gagne, McGovern, Haupt, & Williams, 2017).

Several parameters can be tuned by the grid search method to optimize the performance of the predictive model (Johnson, et al., 2017). One of the parameters is the number of trees that grows sequentially, and another parameter is the depth of the tree that indicates the depth of interaction between features. The learning rate, which is another important parameter of the model, can be tuned to determine how much each tree contributes to the overall performance of the model (Johnson, et al., 2017).

3.2.1.3. Decision tree regression

The decision tree regression is a regression model built on a form of tree-based structures. The model generates predictions on the dependent variable in numeric form (Rathore & Kumar, 2016). The decision tree method can build models with complex variables without having many assumptions on the modeling (Zhao & Zhang, 2008). The method can isolate important independent features by basis function when many variables are used in the model.

The decision tree regression can be unstable, for example a change in the training data can change the output and different attributes for the model need to be selected (Zhao & Zhang, 2008). In this research, the decision tree regression was also applied as it could handle data sets with high dimensionality and could predict a dependent variable in a numeric form (Rathore & Kumar, 2016).

3.4. A Roadmap for Building Machine Learning Predictive Model

Previously, the basic concepts of machine learning, supervised learning, and learning algorithms were discussed. In this section, Figure 6 depicts a workflow diagram for a machine learning predictive modeling which will be discussed below. After acquiring the revenue vehicle inventory data from the NTD database, the initial raw data from the fiscal year 2008 to 2016 were combined and preprocessed for the machine learning algorithm. The preprocessed data were separated into training data with retired vehicles to build the predictive model and deployment data for predictions for retirement. The training data set was split into the training set and the test set. The learning algorithms were applied to the training set to build the predictive model, and various performance measures were applied to the testing set to evaluate the model. After getting the best predictive model, the model was deployed on deployment data for predictions.

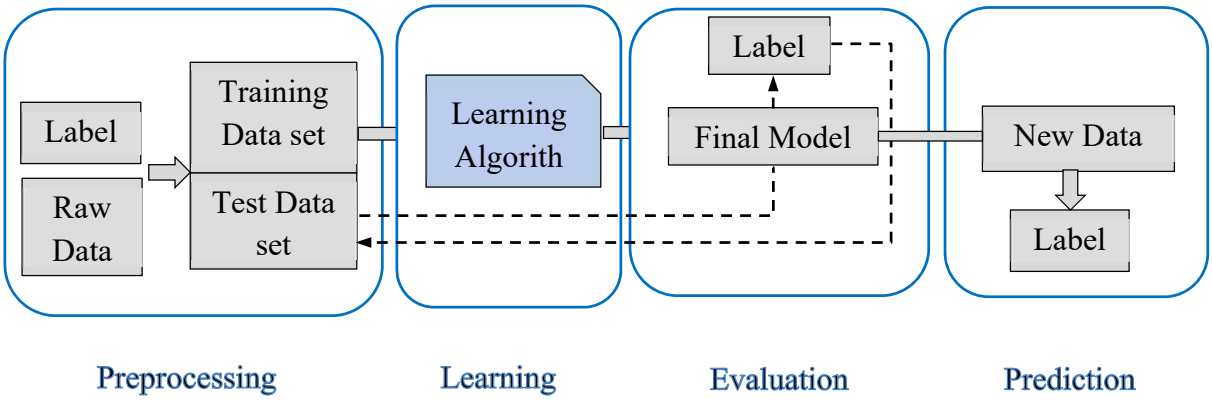


Figure 6. Roadmap for Machine Learning Predictive Model (Adapted from Raschka, Sebastian. 2015. *Python machine learning.* First Edition. Edited by Roshni Banerjee. Birmingham: Packt Publishing Ltd.)

3.5. Preprocessing of Data

The quality of data and the information it contains are key factors of how well a machine learning algorithm can learn. Most of the time, raw data from the source does not come in the form and shape to use in the machine learning algorithm. Therefore, the preprocessing of the data is a critical step before feeding the data to any machine learning application (Raschka, 2015). The NTD databases contain the revenue vehicle inventory data in excel format, which have many general problems related to how transit agencies entered their data and maintained the data structures. In this research, the revenue vehicle inventory data from the fiscal year 2008 to 2016 were processed for a machine learning predictive model to solve the transit state of good repair. The example in Table 9 represents sample data from the vehicle inventory data that were used to build the training data for machine learning algorithms. The columns designate attributes or features which were used to make predictions and the rows designate instances or observations. The first column is called Revenue Vehicle Inventory ID which is unique for each row. The Revenue Vehicle Inventory ID was not used for prediction as it was too specific and pertained to only a single observation (Bowles, 2015).

The attributes shown in Table 9 include numerical and categorical variables. In this example, the numerical variables, Vehicle Length, Manufacture Year, Retired Year, and Service Life, are the most usual type of attributes, whereas Mode, TOS, Vehicle Type, and Fuel Type are categorical variables. These categorical variables were converted to numerical values with either "1," if the category exists, or "0," if it does not exist (Bowles, 2015). Alternatively, the categorical variables could be converted to True or False.

Table 9. Sample Revenue Vehicle Inventory Data

Revenue Vehicle Inventory ID	Mode	TOS	Vehicle Type	Mfr Year	Fuel Type	Vehicle Length	Retired Year	Service Life
53849	VP	DO	Van	2009	Gasoline	17	2014	5
45948	MB	DO	Bus	1995	Diesel Fuel	40	2014	19
24446	DR	PT	Bus	2001	Diesel Fuel	22	2015	14
13756	TB	DO	Trolley Bus	1996	Electric propulsion	37	2013	17

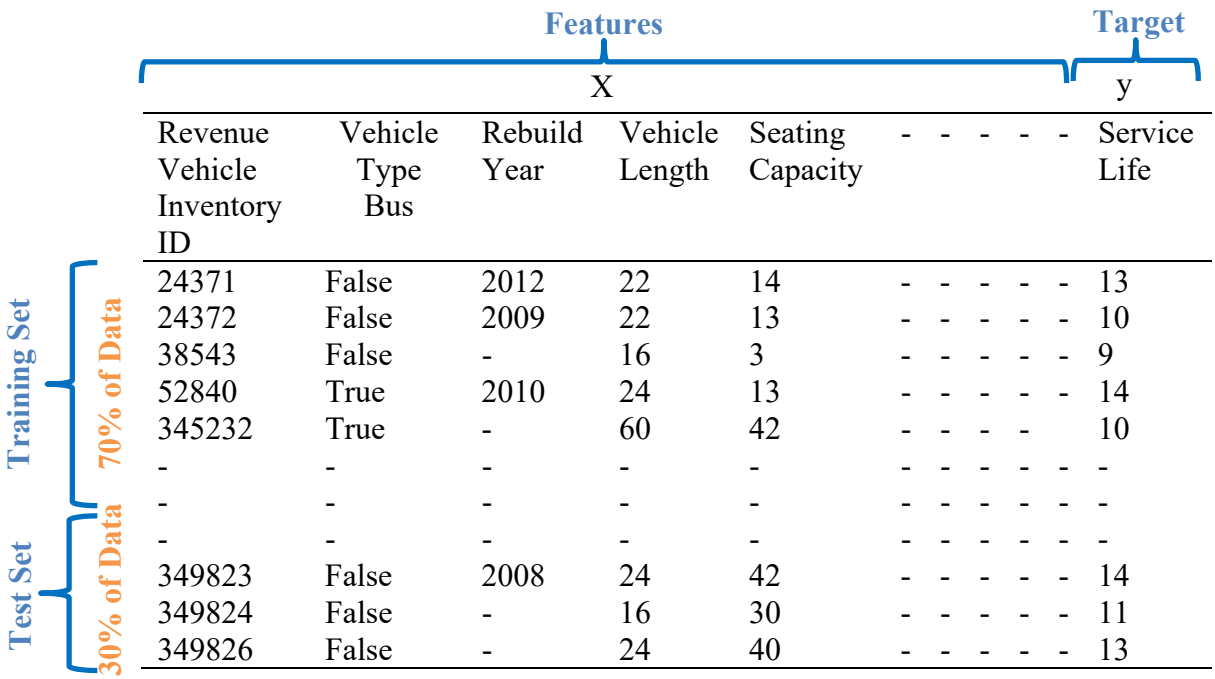
It is common in the real-world application that there might be errors in the data collection process. Therefore, the following items such as data quality, missing records, misspelling of different fuel types or vehicle types, extra whitespaces at the end of the columns, inconsistencies of a column naming in the legacy data sets were taken into consideration to ensure the accuracy of the data. The most common problem is missing values. The missing values were handled either by removing missing entries from the unique vehicle inventory ID or filling missing values in the non-unique attributes with the value calculated by different methods based on data types. In addition, there were misspelling of categorical names or alternate names present in the Fuel Type or Vehicle Type categories. These categorical names were replaced with a normalized form of name to maintain data consistency throughout all the historical data. All of the other issues of

the column names in the historical data were fixed either by replacing or renaming with correct attribute names.

If the Retired column had a Flag Y present, a new column, Retired Year, was created with the value of the year the vehicle was retired. Another new column of Service Life was created with the historical data for training the model. The value of Service Life was generated by subtracting Manufacturing Year from the Retired Year. Since Revenue Vehicle Inventory ID was unique for vehicle identification, it was used for indexing the data sets and in that way, duplication was avoided. The retired vehicles data were used for training and evaluating the model, and the data with the current vehicles in operation were used for predicting the projected service life of the transit vehicles.

3.6. Development of Training Data

In the methodology, the revenue vehicle inventory data sets from the NTD database were used to train the predictive model. The retired revenue vehicles data from 2008 to 2016 that were used as observational data to train and test the predictive model are shown in Figure 7, and the non-retired vehicles' data that are shown in Table 10 were used for predicting the service life of the transit vehicles. At first, the Service Life was calculated from the observational data and was used as target data. Then, the observational data from which the model will learn were split into two separate data sets: the training set, and the testing set. The training set was used for building the model and the testing set was used for evaluation purposes. Here the algorithm or the model will learn from the training data by understanding some correlations to make the prediction, then the models will be evaluated on the testing data.



Revenue Vehicle Inventory ID	Vehicle Type Bus	Rebuild Year	Vehicle Length	Seating Capacity	- - - - -	Service Life
24371	False	2012	22	14	- - - - -	13
24372	False	2009	22	13	- - - - -	10
38543	False	-	16	3	- - - - -	9
52840	True	2010	24	13	- - - - -	14
345232	True	-	60	42	- - - -	10
-	-	-	-	-	- - - - -	-
-	-	-	-	-	- - - - -	-
-	-	-	-	-	- - - - -	-
349823	False	2008	24	42	- - - - -	14
349824	False	-	16	30	- - - - -	11
349826	False	-	24	40	- - - - -	13

Figure 7. Sample Initial Training Set on Revenue Vehicles Data

Table 10. Sample Predictions on Deployment Data After Applying the Predictive Model

Revenue Vehicle Inventory ID	Manufacture Year	Rebuild Year	Vehicle Length	Seating Capacity	- - - - -	Predicted Service Life
54985	2010	-	22	14	- - - - -	12
54986	2009	2014	16	13	- - - - -	14
54987	2014	-	24	50	- - - - -	10
54988	2013	-	30	50	- - - - -	13
54989	2012	-	22	42	- - - -	11
-	-	-	-	-	- - - - -	-
-	-	-	-	-	- - - - -	-

3.7. Parameter Optimization

The regression algorithm requires parameter values to be set up before applying the algorithm. Appropriate parameter settings in the algorithm will provide the best model while bad parameter settings will produce poor results. The best model with the tuned parameter will provide good performance on making predictions on new data with previously unseen values (Ma, 2012). The random forest model works very well without optimizing parameters. However,

the performance of the model can be improved by removing redundant variables, fixing a minimum leaf size, and defining a random state number (Mueller & Massaron, 2015).

In this research, a simple parameter optimization method was used to find the optimal parameters for the random forest regression model. In addition, the grid search methodology was used in the gradient boosting and decision tree regression models to find the optimal parameter values where the points are situated on the grid within the parameter space. The grid search does a complete search starting from the minimum point of the grid in the parameter space to the maximum points and finds the optimal parameters. In short, the grid search chooses the best point after evaluating every point in the grid, and the best value on the best point is considered to be the optimum solution (Ma, 2012).

3.8. Evaluation of Predictive Model

After setting the best parameter values in the model, training the model with regression objects, and fitting the model with the training set of data, the test data set was used to calculate the performance of the model on the unseen data. The performance of the machine learning model was tested by measuring the R^2 score, root mean squared error (RMSE), and mean absolute error (MAE) (Raschka, 2015). Once the evaluation of each model was complete, the performance of each model was compared to each other, and the best performing predictive model was chosen to predict on new data.

RMSE calculates the measure of the model's performance which is simply the square root of the average of the sum of squared error function. In regression problems, RMSE is the primary performance indicator than the other measures for regression problems (Aurlien, 2017). Another performance measure is called mean absolute error (MAE) which was used to check the accuracy of the model's predictions. MAE looks at every prediction the model makes, and it

provides an average mistake across all the predictions (Geitgey, 2017). Another performance measure, the coefficient of determination (R^2) which is the fraction of the response variance, was also used to measure the model performance. The value of R^2 is between 0 and 1, and the model fits the data perfectly if the value is equal to 1.

Figure 8 shows the diagram of machine learning predictive models on the state of good repair which was used to predict the service life of the transit vehicles on the most up-to-date data that the transit agencies had. Using the model, transit agencies will have a clear picture of the condition of their transit vehicles when they will have to retire their revenue vehicles and will help decision makers plan for their SGR estimations.

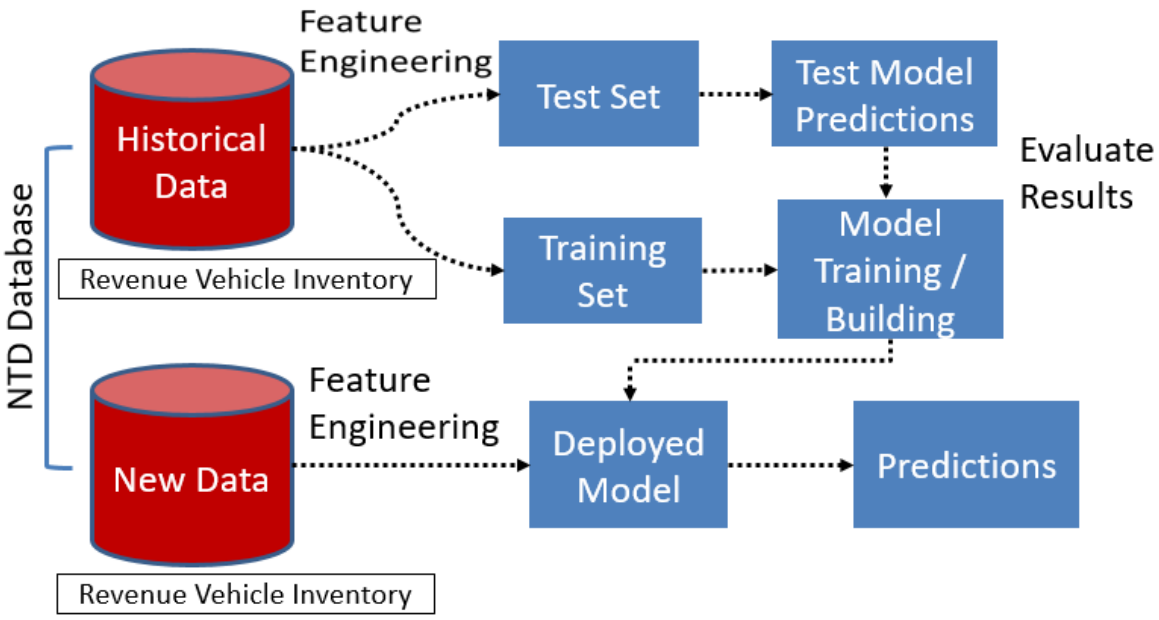


Figure 8. Machine Learning Predictive Model on State of Good Repair

3.9. Summary of Methodology

The methodology involved introducing machine learning techniques to develop a predictive model for the state of good repair to predict the service life of transit vehicles. The methodology discussed on the basic concept of machine learning, the type of machine learning,

and ensemble methods. The regression analysis of the supervised learning was utilized for the problem. The ensemble methods, which are very powerful techniques for machine learning model, were discussed. There are three different machine learning techniques, which were introduced in the methodology; they are random forest regression, gradient boosting regression, and decision tree regression. The random forest regression algorithms create predictions on the individual tree randomly and average them on all trees. The gradient boosting regression trees fit weak models sequentially to the negative gradient of the loss function to minimize the errors on the training set and optimize the predictive model. The decision tree regression is a tree-based structure which generates predictions in a numeric form.

The revenue vehicle inventory data from the NTD database was used to build the predictive model. The preprocessing steps of the data were discussed to format the raw data for machine learning algorithms. Data with retired vehicles were used to train and evaluate the model, and data with non-retired vehicles were used to deploy the trained model for predictions.

The regression analysis requires optimized parameters for the model for the best performance. A grid search method was discussed to find the best parameters. After selecting the best parameters value for each of the three predictive models, three predictive models were built, their performance evaluated, and then compared to each other. Based on the best performance, the gradient boosting regression predictive model was chosen to predict the service life of transit vehicles.

CHAPTER 4. DATA ANALYSIS AND RESULTS

4.1. Exploring the Revenue Vehicle Inventory Data Set

The revenue vehicle inventory data from the NTD database was used for building the predictive model and for performing exploratory data analysis. The NTD is the primary source of information on the transit vehicle systems in the United States. Transit agencies report their transit asset data to the NTD database as a requirement for receiving federal funds from the FTA (FTA, 2017d). Revenue vehicle inventory data sets can be found in XLS format in the NTD database and contain information about revenue vehicles from transit agencies published at the end of each fiscal year. The data sets are available to download from the United States Department of Transportation site at https://www.transit.dot.gov/ntd/ntd-data.

The FTA requires all transit agencies who receive Chapter 53 funds, and use them for public transportation services, to report all transit asset information to the NTD per the FTA's TAM regulation. All transit agencies who also receive 5310 funding for public transportation services must begin reporting to the NTD at the beginning of the 2018 reporting year (FTA, 2017b).

4.2. Tools for Processing the Revenue Vehicle Inventory Data Set for Machine Learning Algorithms

The Python programming language was used to analyze the revenue vehicle inventory data from the NTD database and develop predictive models with machine learning algorithms. Python can be accessed by installing the Anaconda distribution package, which includes the Jupyter Notebook for Python. In this analysis, the older reliable Python 2.7 version was used instead of the latest version (Grus, 2015).

Some additional packages were also used for the analysis, computation, and data visualizations (Grus, 2015). Pandas is a tool that has lot more functionality and provides better performance working and manipulating data sets than Python does. NumPy, a building block of Python that performs the scientific computation, was used for computation. Matplotlib was used to visualize data in the form of bar charts, line charts, and scatterplots. Scikit-learn is a machine learning library in Python. Instead of writing an optimization algorithm, the Scikit-learn library was implemented to build the predictive model (Grus, 2015).

4.3. Data Preprocessing for Initial Training and Deployment Data for Machine Learning Model

The performance of the machine learning model depends on the quality of the data and the information the data set contains. Therefore, it is crucial that the data need to be examined and preprocessed before it can be fed to a learning algorithm.

In order to preprocess the revenue vehicles inventory data sets, a few basic packages for Python were loaded as shown below.

```
# Import file package
import sys
# Import data science packages
import numpy as np
import pandas as pd
```

In addition, the matplotlib and seaborn packages were also imported to visualize the data. The seaborn was used to improve default plot formatting. The inline command *%matplotlib* was used to display all the plots in the iPython Notebook (Hunter, Dale, Firing, & Droettboom, 2017). The block of code is as follows:

```
# Plot pretty figures
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Plot figures inline
%matplotlib inline
```

The following block of code below was used to change the default values and customize the behavior for every plot. The *labelsize* parameters of the axes, as well as the *labelsize* of the xtick and ytick were set a value to adjust the layout (Hunter, Dale, Firing, & Droettboom, 2017).

```
# Set matplotlib parameters in the script
plt.rcParams['axes.labelsize'] = 14
plt.rcParams['xtick.labelsize'] = 12
plt.rcParams['ytick.labelsize'] = 12
```

The default setting of *max_columns* displays 20 columns, and the default setting of *max_info_columns* displays 100 rows per column. If the data frame contains more objects or data points per column, the default setting will truncate the display. Therefore, the settings of the display were changed to '2000' to show all columns and column information in this training data frame (McKinney, Wes; PyData Development Team, 2017). The block of code is as follows:

```
# Set pandas to show all columns and column information in Data Frame
pd.set_option("display.max_columns", 2000)
pd.set_option("display.max_info_columns", 2000)
```

The following block of codes works as a function, was used to save all the figures as PNG format in the root directory under 'images' folder.

```
# Save the figures to a path
ROOT_DIR = "."
IMAGES_PATH = os.path.join(ROOT_DIR, "images")
# Define the function
def save_image(image_name, tight_layout = True, image_extension =
      "png", resolution = 300):
            path = os.path.join(IMAGES_PATH, image_name + "." +
                  image_extension)
            if tight_layout:
                  plt.tight_layout()
plt.savefig(path, format = image_extension, dpi = resolution)
```

The revenue vehicle inventory data sets from fiscal year 2008 to fiscal year 2016 were downloaded from the FTA's NTD database website at https://www.transit.dot.gov/ntd/ntd-data. After downloading data sets to the local drive, the pandas's read_excel() function was applied to all the data. The individual revenue vehicle inventory data was stored in the individual data frame object. A sample block of code is shown below that used 2016 inventory data to read and store data to the data frame, revenue_vehicle_inventory_16. The other data frames from years 2008 to the 2015 data were created in a similar way to that of the year 2016 data set.

```
# Read annual revenue vehicle inventory data of the fiscal year 2016
revenue_vehicle_inventory_16 = pd.read_excel('..//NTD/2016/Revenue
      Vehicle Inventory_0.xlsx')
```

A data frame is a rectangular table of data which contains columns of different value types such as numeric, string, or Boolean, etc. The data in the data frame is stored as one or more two-dimensional blocks rather than a list or some other collection of one-dimensional arrays (McKinney, 2017). A new data frame was created that indicated what columns needed to be included in the data frame for further feature engineering.

```
# Select columns for models
df_all = pd.DataFrame(columns = ['NTD ID', 'Agency Name', 'Mode',
      'TOS', 'Revenue Vehicle Inventory ID', 'Total Fleet Vehicles',
      'Dedicated Fleet', 'Vehicle Type', 'Ownership Type', 'Funding
      Source', 'Manufacture Year', 'Rebuild Year', 'Manufacturer',
      'Model', 'Active Fleet Vehicles', 'ADA Fleet Vehicles',
      'Emergency Contingency Vehicles', 'Fuel Type', 'Vehicle Length',
      'Seating Capacity', 'Standing Capacity', 'Total Miles on Active
      Vehicles During Period', 'Average Lifetime Miles per Active
      Vehicles', 'Supports Mode', 'Supports Service', 'Retired',
      'Retired Year'])
```

A function *append_to_frame()* was defined that added data from previous years to the existing data. The code is as follows:

```
# Define a function which add data from previous years
def append_to_frame(appendee, appender, match):
      add_to = appender.loc[~appender[match].isin(appendee[match]), :]
      return appendee.append(add_to)
```

The individual data frame for each year needed to be cleaned up individually before combined into a single data frame. Since data were entered into the spreadsheet without following any guidelines, the data sets were not consistent from year to year. For example, column names were mismatched in many data sets from year to year. Therefore, the column names were fixed by renaming or removing some unnecessary columns. Some data points were dropped because they had a null inventory ID. There were also whitespaces that existed in the categorical columns, which were fixed by removing extra whitespaces. The following examples show how the block of codes was used to rename columns and drop unnecessary columns.

```
# Rename columns
revenue_vehicle_inventory_16 =
      revenue_vehicle_inventory_16.rename(columns = {'5 Digit NTD ID':
      'NTD ID'})
# Drop unnecessary columns
revenue_vehicle_inventory_16 =
      revenue_vehicle_inventory_16.drop(['Legacy NTD ID', 'Reporting
      Module', 'Reporter Type', 'Other Manufacturer Description'], axis
      = 1)
```

A new column Retired Year was added to each data frame based on the information on column Retired = Y. The following block of code was used to create the Retired Year column and added value by inserting 2016.

```
# Created new columns 'Retired Year'
revenue_vehicle_inventory_16['Retired Year'] =
      np.where(revenue_vehicle_inventory_16['Retired'] == 'Y', '2016',
      '')
# Fill NaN with 'N' in 'Retired' field
```

```
revenue_vehicle_inventory_16['Retired'].fillna('N', inplace = True)
```

After the initial cleanup of individual data, the following code was used to add data to the previous data frame. Since Revenue Vehicle Inventory ID is unique, it prevents duplicating data. The code below was used to add the 2016 inventory data to the initial blank data frame.

```
# Add revenue vehicle inventory data for 2016 to the blank Data Frame
df_all = append_to_frame(df_all, revenue_vehicle_inventory_16, 'Revenue
      Vehicle Inventory ID')
```

Since the revenue vehicle inventory data from 2008 to 2016 were used, each data set needed to be cleaned up separately before adding it to the combined data frame. In this analysis, only the data cleaning procedure on 2016 revenue vehicle inventory data is demonstrated here. For the remaining data sets, additional cleaning procedures may have been required depending on the quality of the data. After the initial cleanup of all the data sets, the remaining data sets from the years 2008 to 2015 were combined to the 2016 data set and stored into a data frame. The following example shows how the sample block of codes was used to combine 2015 data set with the 2016 data frame.

```
# Add inventory data from 2015 to the previous Data Frame (2016)
df_all = append_to_frame(df_all, revenue_vehicle_inventory_15, 'Revenue
      Vehicle Inventory ID')
```

The following code shows the number of rows and columns in the combined data frame.

```
# Check the number of rows and columns
df_all.shape
```

The above code showed 42440 rows and 27 columns in the initial combined data frame.

Four categorical data types, called Fuel Type, Vehicle Type, Funding Source, and Ownership Type, had categorical names. These categorical names contained whitespaces at the

end of the names. The following code was used to clean whitespaces from the Fuel Type categorical name. The other categorical names mentioned above were cleaned in a similar way.

```
# Remove whitespaces from the categorical name
df_all['Fuel Type'] = df_all['Fuel Type'].str.rstrip()
```

Initially, some data cleaning and manipulation were performed before combining all the data frames. Then, the *info()* method was used in order to see important information about the full inventory data frame, such as the number of data points, data columns, and data type stored in each column. This information indicated which columns were numeric or strings and whether or not all columns had complete data points in them. The *df_all.info()* command displayed the basic information about the data frame, which is listed in Table 11. The information below in Table 11 shows that there is missing information in the data set that might have caused problems if not fixed before the model was built.

Table 11. Data Columns Information Table

Data Columns	Data Points	Data Value	Data Type
ADA Fleet Vehicles	41104	non-null	float64
Active Fleet Vehicles	42401	non-null	float64
Agency Name	42440	non-null	object
Average Lifetime Miles per Active Vehicles	24177	non-null	float64
Dedicated Fleet	42422	non-null	object
Emergency Contingency Vehicles	17579	non-null	float64
Fuel Type	27616	non-null	object
Funding Source	42416	non-null	object
Manufacture Year	39251	non-null	float64
Manufacturer	25539	non-null	object
Mode	42440	non-null	object
Model	25401	non-null	object
NTD ID	42440	non-null	object
Ownership Type	42418	non-null	object
Rebuild Year	2201	non-null	float64
Retired	32106	non-null	object
Retired Year	42440	non-null	object
Revenue Vehicle Inventory ID	42422	non-null	object
Seating Capacity	42406	non-null	float64
Standing Capacity	23176	non-null	float64
Supports Mode	5983	non-null	object
Supports Service	6087	non-null	object
TOS	42440	non-null	object
Total Fleet Vehicles	42419	non-null	object
Total Miles on Active Vehicles During Period	24147	non-null	float64
Vehicle Length	39287	non-null	float64
Vehicle Type	42421	non-null	object

The following block of code calculated the missing information in each column.

```
# Calculate total missing values
total = df_all.isnull().sum().sort_values(ascending = False)
# Convert missing values to percentage
percent =
      (df_all.isnull().sum()/df_all.isnull().count()).sort_values(ascen
      ding = False)*100
missing_data = pd.concat([total, percent], axis = 1, keys = ['Total
      Missing Data', 'Percent of Missing Data'])
```

The output is shown in Table 12. The table below shows that the data were missing in different data types. Therefore, different approaches were initiated to fill in the missing data.

Table 12. Missing Data Information

Columns	Total Missing Data	Percent of Missing
Rebuild Year	40239	94.813855
Supports Mode	36457	85.902451
Supports Service	36353	85.657399
Emergency Contingency Vehicles	24861	58.579171
Standing Capacity	19264	45.391140
Total Miles on Active Vehicles During Period	18293	43.103205
Average Lifetime Miles per Active Vehicles	18263	43.032516
Model	17039	40.148445
Manufacturer	16901	39.823280
Fuel Type	14824	34.929312
Manufacture Year	3189	7.514138
Vehicle Length	3153	7.429312
ADA Fleet Vehicles	1336	3.147974
Active Fleet Vehicles	39	0.091894
Seating Capacity	34	0.080113
Funding Source	24	0.056550
Ownership Type	22	0.051838
Total Fleet Vehicles	21	0.049482
Vehicle Type	19	0.044769
Revenue Vehicle Inventory ID	18	0.042413
Dedicated Fleet	18	0.042413
Mode	0	0.000000
NTD ID	0	0.000000
Retired	0	0.000000
Retired Year	0	0.000000
TOS	0	0.000000
Agency Name	0	0.000000

At this point, for further analysis a copy of the combined data was saved as a CSV file by executing the following code:

```
# Save all data to a comma separated (CSV) file
df_all.to_csv('Revenue_Vehicle_Inventory_all_years.csv', sep = ',')
```

In a pandas data frame, the index is a special column that contains the row labels (Downey, 2014). Since the Revenue Vehicle Inventory ID column was unique, the column was set as the index. The code is as follows:

```
# Set 'Revenue Vehicle Inventory ID' as index
df_all = df_all.set_index('Revenue Vehicle Inventory ID')
```

Since the Manufacture Year is vital for calculating the service life of the vehicle, it is important that the data must include the Manufacture Year. However, the calculation of missing data estimated that the Manufacture Year data showed 3189 null data points. Since interpolation techniques cannot fill these missing values in the Manufacture Year, this huge number of important data points were removed by running the following code:

```
# Drop rows if Manufacture Year is missing
df_all.dropna(subset = ['Manufacture Year'], inplace = True)
```

4.3.1. Removing unnecessary columns

Some variables from the data sets were not required for either data analysis or modeling. Therefore, the *drop()* method was applied to remove unnecessary columns using the following code (Mueller & Massaron, 2015):

```
# Remove columns
df_all = df_all.drop(['Agency Name', 'NTD ID', 'Manufacturer', 'Model',
      'Retired', 'Supports Service'], axis = 1)
```

4.3.2. Dealing with missing data

There could have been many reasons that the real-world applications may have had missing values during the data collection process. Sometimes, some fields are left blank as NaN (Not a Number) in the database. Unfortunately, machine learning algorithms cannot handle missing values. Thus, it is very important to take care of the missing values before analyzing and

modeling. Since the inventory data frame is large, it would be tedious to look for missing values. Therefore, the *isnull()* method was used in the data frame to see if the column contained missing values with *True* and numeric values with *False*. Finally, the *sum()* method was used to calculate the total number of missing values per column. The code is as follows:

```
# Check number of nulls
df_all.isnull().sum()
```

The output is shown in Table 13.

Table 13. Number of Null Points in the Columns

Column Names	Number of Null Points
ADA Fleet Vehicles	1207
Active Fleet Vehicles	10
Average Lifetime Miles per Active Vehicles	15121
Dedicated Fleet	0
Emergency Contingency Vehicles	21719
Fuel Type	14100
Funding Source	5
Manufacture Year	0
Mode	0
Ownership Type	3
Rebuild Year	37050
Retired Year	0
Seating Capacity	9
Standing Capacity	16118
Supports Mode	33523
TOS	0
Total Fleet Vehicles	2
Total Miles on Active Vehicles During Period	15151
Vehicle Length	3
Vehicle Type	0

4.3.3. Filling in missing data

Missing data is common in most of the data analysis. Rather than filtering out missing data, it can be filled in many ways. However, in this case, the missing values (NaN) in arithmetic operation were filled in with either applying the constant value of "0" or applying an appropriate

function. On the other hand, the categorical values were filled with prefixing “Unknown” followed by the category name (McKinney, 2017). The block of codes for arithmetic filling and categorical filling are as follows (only one of each sample column was shown here):

For arithmetic filling:

```
# Fill NaN values with zero (0)
df_all['ADA Fleet Vehicles'] = df_all['ADA Fleet Vehicles'].fillna(0)
```

For categorical filling:

```
# Fill NaN values with 'Unknown' followed by category name
df_all['Fuel Type'] = df_all['Fuel Type'].fillna('Unknown Fuel')
```

Furthermore, the Vehicle Length and the Seating Capacity categorical fields were filled by averaging with the *mean()* function as follows.

```
# Fill NaN values with mean() function
df_all['Vehicle Length'].fillna(value = df_all['Vehicle
      Length'].mean(), inplace = True)
df_all['Seating Capacity'].fillna(value = df_all['Seating
      Capacity'].mean(), inplace = True )
```

And finally, the missing values in Support Mode was filled by Mode by the following code:

```
# Fill NaN with values from 'Supports Mode'
df_all['Supports Mode'].fillna(value = df_all['Mode'], inplace = True)
```

Once missing values were filled, the following code verified whether there were any missing values remain in the data frame.

```
# Check number of null data points in each column
df_all.isnull().sum()
```

The output is shown in Table 14. A zero (0) in each column indicates no missing values. The table shows the number of missing values.

Table 14. Number of Data Points in Each Column

Column Names	Number of Null Data Points
ADA Fleet Vehicles	0
Active Fleet Vehicles	0
Average Lifetime Miles per Active Vehicles	0
Dedicated Fleet	0
Emergency Contingency Vehicles	0
Fuel Type	0
Funding Source	0
Manufacture Year	0
Mode	0
Ownership Type	0
Rebuild Year	0
Retired Year	0
Seating Capacity	0
Standing Capacity	0
Supports Mode	0
TOS	0
Total Fleet Vehicles	0
Total Miles on Active Vehicles During Period	0
Vehicle Length	0
Vehicle Type	0

4.3.4. Clean up of categorical names

Since legacy data from 2008 through 2016 were used in the data set, there were naming inconsistencies in categorical columns called Fuel Type, Vehicle Type, Funding Source, and Ownership Type. Therefore, the *replace()* function was applied to rename the inconsistent names of those categorical columns. The following is a sample code of renaming categorical names in Fuel Type:

```
# Rename fuel type for consistencies
df_all['Fuel Type'].replace({'Bio-diesel(BD)': 'Diesel Fuel', 'Bunker
      fuel': 'Diesel Fuel', 'Compressed natural gas (CNG)': 'Compressed
      Natural Gas', 'Diesel fuel': 'Diesel Fuel', 'Diesel
      Fuel/Liquefied Petroleum Gas': 'Diesel Fuel', 'Dual fuel':
      'Diesel Fuel/Compressed Natural Gas', 'Electric battery':
      'Electric Battery', 'Electric propulsion': 'Electric Propulsion
      Power', ‘Gasoline/Compressed Natural Gas': 'Gasoline',
```

```
'Gasoline/Ethanol': 'Gasoline', 'Hybrid diesel': 'Hybrid Diesel',
'Hybrid gasoline': 'Hybrid Gasoline', 'Hybird gasonline': 'Hybrid
Gasoline', 'Hybrid Gasoline/Ethanol': 'Gasoline', 'Hybrid
Gasoline/Liquefied Petroleum Gas': 'Gasoline', 'hydrogen (HY)':
'Hydrogen Cell', 'Liquefied natural gas (LNG)': 'Liquefied
Natural Gas', 'Liquefied petroleum gas (LPG)': 'Liquefied
Petroleum Gas', 'Other (specify in box below)': 'Other'}, inplace
= True)
```

The renaming of categorical names from Vehicle Type, Funding Source, and Ownership Type were renamed in a similar manner.

4.3.5. Create the initial training data

The development of algorithms starts with building training sets. The training set consists of the two types of data, such as the target data and the features for making the prediction (Bowles, 2015). In order to create the training set, retired vehicles were filtered out of the data from 2008 to 2016. The following code generated the training set:

```
# Filter data which have been retired since 2008 until 2016
df = df_all.loc[df_all['Retired Year'].isin(['2016', '2015', '2014',
      '2013', '2012', '2011', '2010', '2009', '2008'])]
```

The above code indicated that the training data with Retired Year were stored in a new data frame *df*. The target column Service Life was created by subtracting Manufacture Year from Retired Year by executing the following code:

```
# Create new column by subtracting 'Manufacture Year' from 'Retired
# Year'
df['Service Life'] = df[['Retired
      Year']].astype(float).sub(df['Manufacture Year'], axis = 0)
```

Since the target column was created from Retired Year, which was no longer needed in the training set, the column was removed from the training set by executing the following code:

```
# Drop column 'Retired Year'
df = df.drop(['Retired Year'], axis = 1)
```

After initial analysis of the newly created column Service Life, some very low service life figures were found, and in some cases, negative service life figures were found. These negative service life or low service life figures were caused by inaccurate inputting in either the manufactured year or the retired year. Therefore, the training data was further removed from the data that had the service life field with either 0 or -1 values by the following code:

```
df = df.drop(df[df['Service Life'] == -1].index)
df = df.drop(df[df['Service Life'] == 0].index)
df.shape
```

After the drop function removed 330 data points from the training data, the shape attribute showed 7772 total data points in the training data.

At this stage, the initial training data were saved for further preprocessing for machine learning algorithms. The data were saved in a CSV format in the same folder as the iPython Notebook by executing the following code:

```
# Save initial training data for further processing
df.to_csv('initial_training.csv', sep= ',')
```

4.3.6. Create the initial deployment data

Since the training data were created based on the Retired Year column from 2008 to 2016, the rest of the data points did not have any values in the Retired Year column. Therefore, the initial deployment data set was created by filtering out data that were not retired; this operation was performed by logically negating the training data frame. The following code filtered out non-retired vehicles data and stored them in a new data frame:

```
# Filter data which have not been retired since 2008 until 2016
df_not_retired = df_all.loc[~df_all['Retired Year'].isin(['2016',
      '2015', '2014', '2013', ‘2012’, '2011', '2010', '2009', '2008'])]
```

Since there was no need to create a target column in the deployment data, the Retired Year was no longer needed and was removed from the deployment data by executing the following code:

```
# Drop unnecessary column 'Retired Year'
df_not_retired = df_not_retired.drop(['Retired Year'], axis = 1)
```

The following code showed the number of data points in the deployment data:

```
# Check the number of rows and columns of the non-retired vehicles data
df_not_retired.shape
```

The above code showed 31149 vehicles in the deployment set for which the predictive model was used to predict when the vehicles needed to be retired from service. At this stage, the initial deployment data set was saved in a CSV format in the same folder as the current iPython Notebook by executing the following code:

```
# Save the non-retired vehicle data for further processing
df_not_retired.to_csv('NonRetired_Revenue_Vehicle_Data_from_2008_to_201
      6.csv', sep = ',')
```

4.4. Analyzing Important Characteristics of Revenue Vehicle Inventory Training Data Set

Exploratory data analysis is the first step of analysis before creating a training data set for a machine learning model (McKinney, 2017). Because the revenue vehicle inventory data were used to build the training set for the predictive model, it was also important to analyze this data to see the significant value of the model. The following code created a new data frame called *df_analysis* by renaming two columns for easy manipulation to analyze the training data:

```
# Rename columns for easy manipulation
df_analysis = df.rename(columns = {'Vehicle Type': 'Vehicle_Type',
      'Service Life': 'Service_Life'})
```

Previously, there were 7772 vehicles data were found in the training data. Now, the *value_counts()* function was used to calculate the number of vehicles in the training data in each vehicle type.

```
# Count the number of vehicles by vehicle type
df_analysis.Vehicle_Type.value_counts()
```

The output is shown in Table 15. The information about the training data set showed that there was a large number of buses and vans available to train the model compared to other vehicle types. The Double Decker Bus and Inclined Plane Vehicle each have a single data point, which may not be a good fit for the model for these vehicle types.

Table 15. Number of Vehicles by Vehicle Type

Vehicle Type	Number of Vehicle
Bus	3992
Van	2236
Cutaway	719
Automobile	232
Minivan	93
Commuter Rail Passenger Coach	89
Over-the-road Bus	84
Articulated Bus	72
Ferryboat	55
Commuter Rail Locomotive	45
Commuter Rail Self-Propelled Passenger Car	40
Heavy Rail Passenger Car	33
Sports Utility Vehicle	21
Light Rail Vehicle	15
Vintage Trolley	10
School Bus	7
Cable Car	4
Trolleybus	3
Inclined Plane Vehicle	1
Double Decker Bus	1

The vehicle type group data was further visualized by plotting a bar graph by executing the following code:

```
# Count number of vehicles
df_analysis.Vehicle_Type.value_counts().sort_values().plot(kind =
      'barh', figsize = (14,5))
plt.title('Number of Vehicles per Vehicle Type')
plt.xlabel('Number of Vehicles')
plt.ylabel('Vehicle Type')
```

The above code plotted the number of vehicles by vehicle type as a bar plot shown in Figure 9.

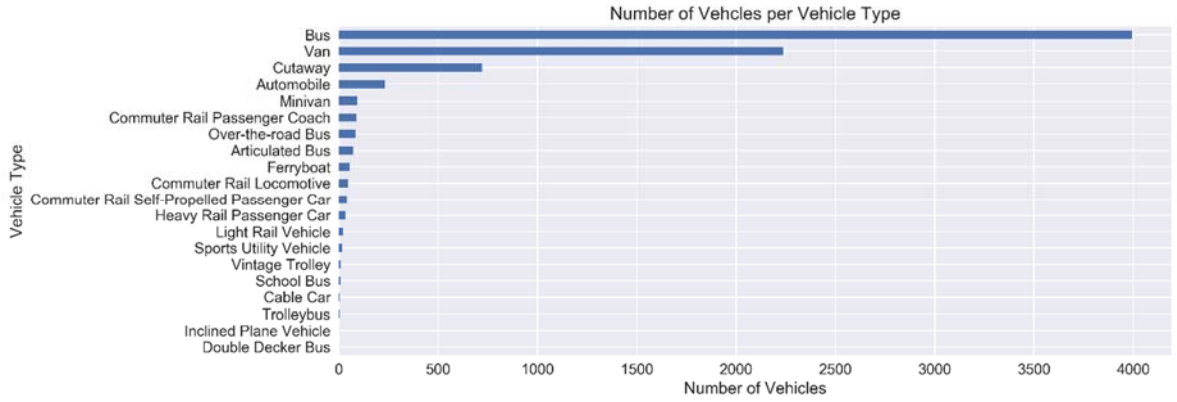


Figure 9. Bar Plot of Number of Vehicles by Vehicle Type

Summarizing target values in the training data can be very useful. The *agg()* function was used to view some typical summary statistics of the mean, standard deviation, minimum value, maximum value, and element counts on the target variable by category. The code is as follows:

```
# Statistical Analysis of Service Life by vehicle type
df_analysis.groupby('Vehicle_Type').Service_Life.agg(['count', 'min',
      'max', 'mean', 'std'])
```

The output of the statistical summary is shown in Table 16. The statistical analysis showed a clear picture of the training data.

Table 16. Statistical Analysis of Service Life by Vehicle Type

Vehicle Type	count	min	max	mean	std
Articulated Bus	72	5	22	12.361	3.358
Automobile	232	5	19	8.495	2.727
Bus	3992	8	29	12.759	3.229
Cable Car	4	49	105	87.25	25.747
Commuter Rail Locomotive	45	18	44	30.511	8.988
Commuter Rail Passenger Coach	89	18	65	41.898	10.037
Commuter Rail Self-Propelled Passenger Car	40	24	51	37.625	6.739
Cutaway	719	7	21	9.965	2.142
Double Decker Bus	1	63	63	63	-
Ferryboat	55	18	94	40.09	13.502
Heavy Rail Passenger Car	33	18	47	29.151	6.562
Inclined Plane Vehicle	1	135	135	135	-
Light Rail Vehicle	21	24	84	41.857	18.65
Minivan	93	5	18	7.946	2.810
Over-the-road Bus	84	5	22	12.476	4.105
School Bus	7	9	22	16.571	5.028
Sports Utility Vehicle	15	5	24	9.733	5.391
Trolleybus	3	13	22	16.666	4.725
Van	2236	6	16	8.482	1.875
Vintage Trolley	10	9	99	59	26.284

Sometimes, the data can be more useful for analysis if it can be visualized it in a plot. Therefore, a horizontal bar plot was drawn by vehicle type on the mean value of the Service Life target feature. The code is as follows:

```
# Plot Statistical Analysis of service life by vehicle type
df_analysis.groupby('Vehicle_Type').Service_Life.agg(['mean']).plot(kin
      d = 'barh', figsize = (14, 5));
plt.title("Mean Service Life by Vehicle Type")
plt.xlabel("Service Life")
plt.ylabel("Vehicle Type")
```

The bar plot with the mean value of Service Life is shown in Figure 10.

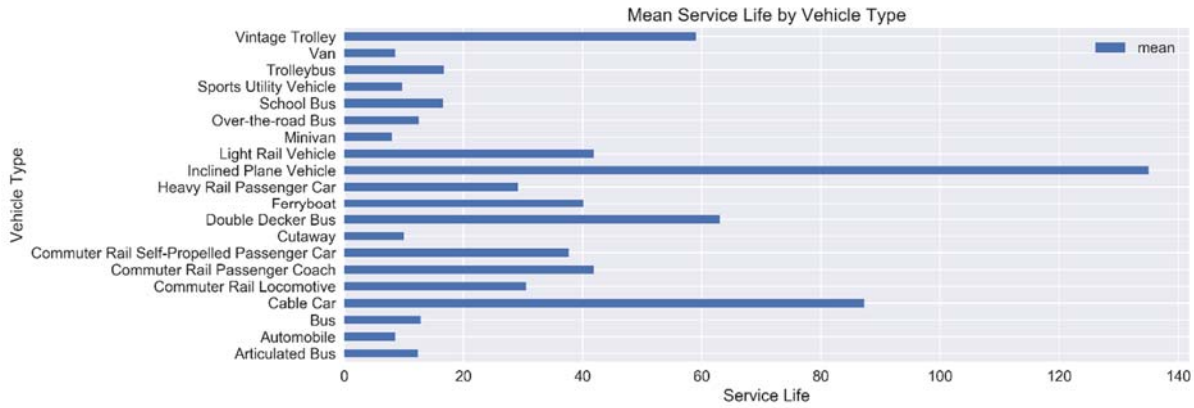


Figure 10. Bar Plot with the Mean Value of Service Life

The contingency Table 17 displays a relationship between qualitative variables by matching two different categorical distributions. The *crosstab()* function in pandas matches variables and identify relationships (Mueller & Massaron, 2015). A contingency table was created between Fuel Type and Mode by executing the following code:

```
# Create cross tabulation on fuel type by vehicle mode
pd.crosstab(df_analysis['Fuel Type'], df_analysis.Mode, margins = True)
```

The contingency table between Fuel Type and Mode is shown in Table 17. The contingency table shows us the tally of how many vehicles belong to each combination of fuel type and mode and that particular fuel types and modes never appear together.

Table 17. Contingency Table Between Fuel Type and Mode

Fuel Type	Mode															All
	AR	CB	CC	CR	DR	DT	FB	HR	IP	LR	MB	RB	SR	TB	VP	YR	
Compressed Natural Gas	0	12	0	0	54	0	0	0	0	0	146	1	0	0	5	0	218
Diesel Fuel	3	127	0	33	1063	0	51	0	0	0	2119	0	0	0	9	1	3406
Diesel Fuel/Compressed Natural Gas	0	0	0	2	13	0	0	0	0	0	51	0	0	0	1	0	67
Diesel Fuel/Electric Propulsion Power	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
Dual Fuel	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	2
Electric Battery	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	15
Electric Propulsion Power	0	0	4	48	0	0	0	33	1	22	0	0	9	3	0	0	120
Ethanol	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	2
Gasoline	0	16	0	0	1662	0	4	0	0	0	325	0	0	0	801	0	2808
Gasoline/Liquefied Petroleum Gas	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	4
Hybrid Diesel	0	0	0	0	1	0	0	0	0	0	30	1	0	0	0	0	32
Hybrid Gasoline	0	0	0	0	9	0	0	0	0	0	3	0	0	0	0	0	12
Hydrogen Cell	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	2
Liquefied Natural Gas	0	0	0	0	0	0	0	0	0	0	32	0	0	0	0	0	32

Table 17. Contingency Table Between Fuel Type and Mode (continued)

Fuel Type	Mode															All
	AR	CB	CC	CR	DR	DT	FB	HR	IP	LR	MB	RB	SR	TB	VP	YR	
Liquefied Petroleum Gas	0	0	0	0	5	0	0	0	0	0	16	0	0	0	0	0	21
Other	0	0	0	0	1	0	0	0	0	0	3	0	0	0	0	0	4
Unknown Fuel	2	61	0	85	600	5	0	0	0	0	241	3	0	0	9	0	1006
All	5	216	4	168	3415	5	55	33	1	22	2984	6	9	3	825	1	7752

4.5. Visualizing Important Characteristics of Revenue Vehicle Inventory Training Data Set

Before training a machine learning model, it is very important to perform an exploratory data analysis on training data to visually detect outliers, distribution of the data, and relationships between features. A scatterplot matrix was plotted to visualize the correlations between features. A *pairplot()* function was used to plot scatterplot from python's seaborn library based on matplotlib (Mirjalili & Raschka, 2017). The code is as follows:

```
# Rename columns for easy manipulation
df_scatter = df_analysis.rename(columns = {'Vehicle Length': 'VL',
      'Seating Capacity': 'SC', 'Standing Capacity': 'STC',
      'Service_Life': 'SL'})
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set(font_scale = 1.3)
cols = ['VL', 'SC', 'STC', 'SL']
g = sns.PairGrid(df_scatter[cols], size = 3, aspect = 1.5)
g.map(plt.scatter)
save_image('scatterplot')
plt.show()
```

The scatterplot matrix provided a graphical summary of feature relationships in the training data set (Mirjalili & Raschka, 2017) and is shown in Figure 11.

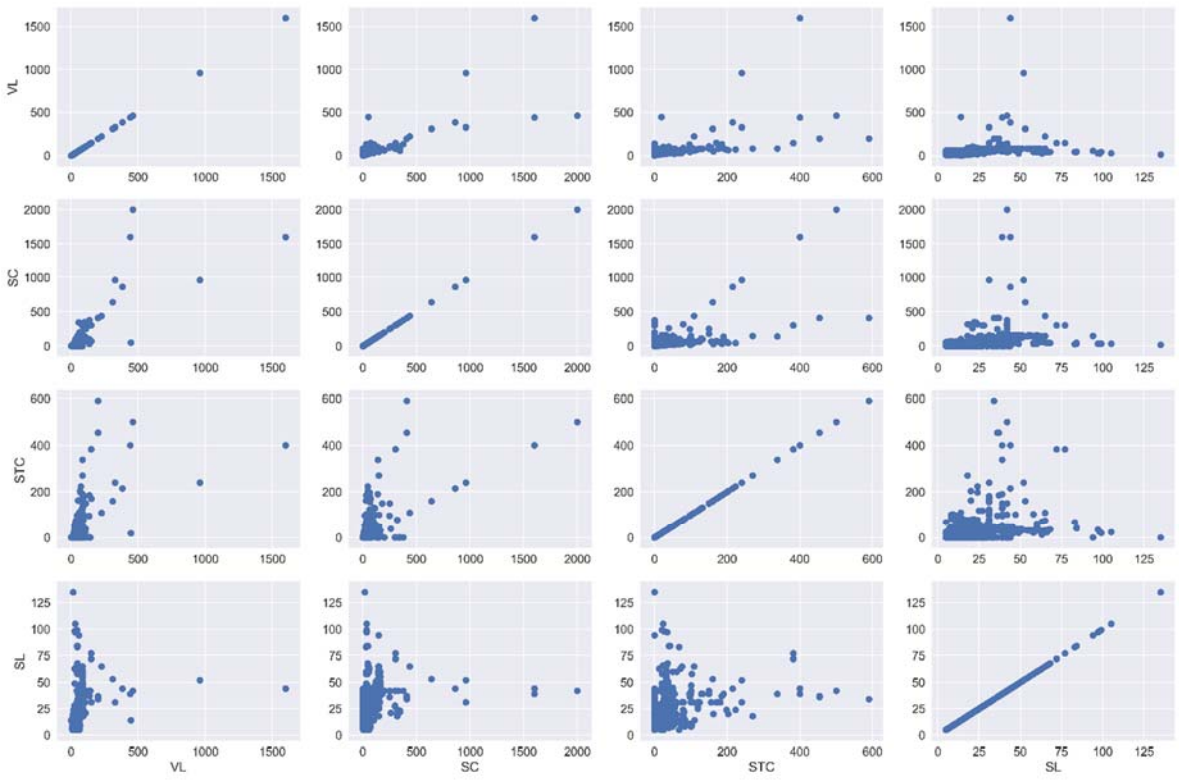


Figure 11. Scatterplot to Visualize the Correlation Amongst Internal Features

Due to readability, a few of the columns from the training data set were used to plot scatterplot. They are Vehicle Length (VL), Standing Capacity (STC), Seating Capacity (SC), and Service Life (SL). By visualizing this scatterplot matrix, data distribution can be analyzed, and outliers can be detected very easily. The above scatterplot showed that even though it had outliers, it distributed normally. However, there is no strong linear relationship between any two features.

Furthermore, visualizing interrelationships between variables and a target feature using scatter plot can be a very useful way to explore the relationship between two attributes. It can show patterns in the data and so data belong to certain groups and data outside of the expected range can be easily visualized (Mueller & Massaron, 2015). The block of code created a scatter plot between target feature (Service Life) and a numerical feature (Vehicle Length).

In the below scatter plot, most of the points lie in a group and tend to form a straight line, but some of the points scatter around the plot. Therefore, these two variables are somehow linearly, but not strongly, correlated. While visualizing the graphs can help explore the data for patterns, some outliers in the data upon further analysis can be explained. For example, after further analysis, it seemed these seemingly outlier points were actually for the Ferry Boat category, which had a longer service life.

```
# Scatter plot of Vehicle Length vs. Service Life
var = 'Vehicle Length'
data = pd.concat([df_analysis['Service_Life'], df_analysis[var]], axis
      = 1)
data.plot.scatter(y = var, x = 'Service_Life', ylim = (0, 1650),
      figsize = (12, 5))
plt.xlabel('Service Life')
plt.title('Service Life vs. Vehicle Lenght')
save_image('service_life_hist')
plt.show()
```

The above code plotted the scatter plot shown in Figure 12.

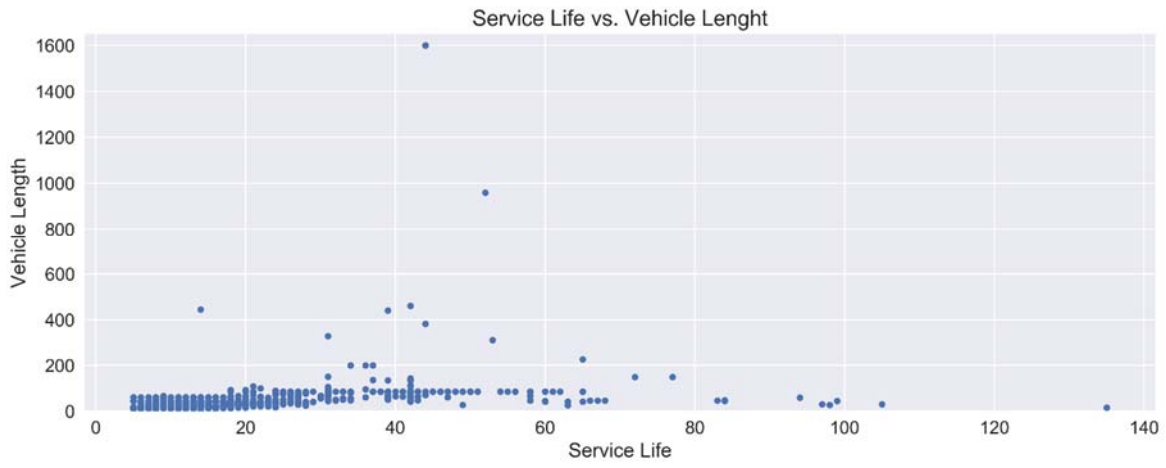


Figure 12. Scatter Plot of Service Life vs. Vehicle Length

Two additional scatter plots were created in the same general procedure described above in order to visualize relationships. One of the scatter plots was created with the target variable Service Life and numerical variable Seating Capacity, as shown in Figure 13. The other scatter plot was created with Service Life and Standing Capacity shown in Figure 14. In the below scatter plots, there are no strong linear correlations between Service Life versus either Seating Capacity or Standing Capacity. Therefore, the data would be a good fit for the nonlinear regression model instead of the linear regression model.

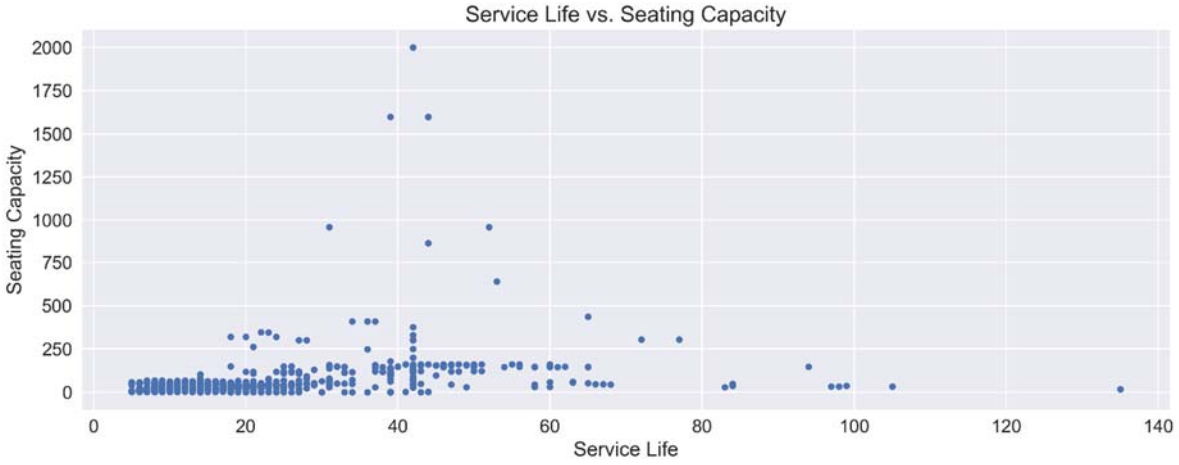


Figure 13. Scatter Plot of Service Life vs. Seating Capacity

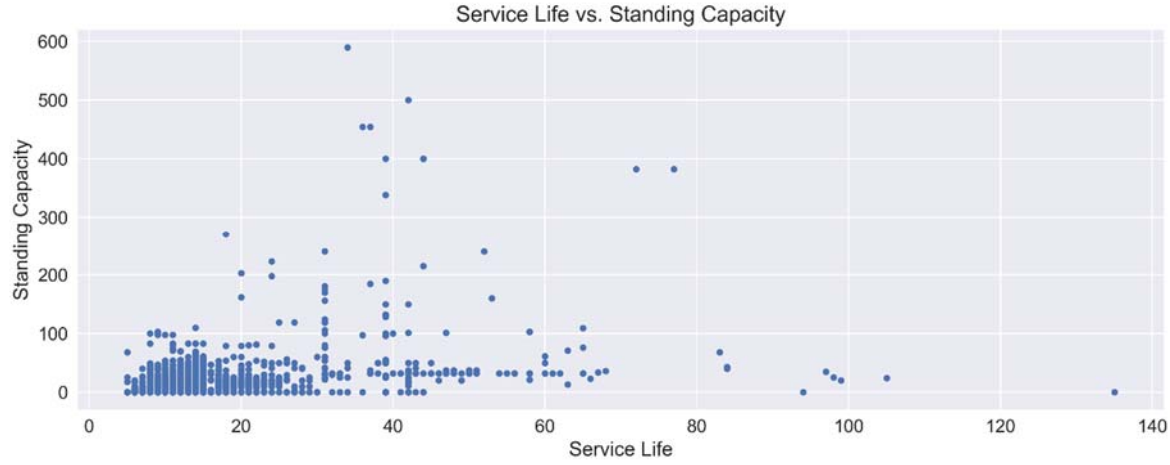


Figure 14. Scatter plot of Service Life vs. Standing Capacity

4.6. Visualizing Relationships Between a Target Feature and Categorical Features

Box plots are an effective way to visualize a numeric column across several categories and to provide statistical summaries. Box plots use rectangles with internal lines that show the median value (also called 50th percentile) for the column. In addition, each rectangle also has two horizontal external lines attached by a vertical line to the top and bottom of the box, which indicate the 75th percentiles and 25th percentiles, respectively. The box, or the rectangle itself, contains the values in the interquartile range between the 75th and the 25th percentile of the data. The data points below and above the limit indicate outliers (Downey, 2014). The following block of code was used to create a box plot on target variable Service Life by the categorical variable Vehicle Type.

```
# Boxplot of service life by vehicle type
var = 'Vehicle_Type'
data = pd.concat([df_analysis['Service_Life'], df_analysis[var]], axis
      = 1)
f, ax = plt.subplots(figsize = (18, 11))
fig = sns.boxplot(x = var, y = "Service_Life", data = data, palette =
      "Set3")
ax.set_title('Service Life vs Vehicle Type')
plt.xlabel('Vehicle Type')
plt.ylabel('Service Life')
fig.axis(ymin = 0, ymax = 140)
plt.xticks(rotation = 45);
save_image('boxplot_vt')
```

Figure 15 shows a box plot of Service Life versus Vehicle Type. The dots in the box plot indicate service life outliers in each vehicle type. These outliers may reduce the performance of the model. However, some machine learning algorithms can handle these outliers effectively and provide a good predictive model. However, the performance of the predictive model can be optimized by eliminating outliers from the training data set.

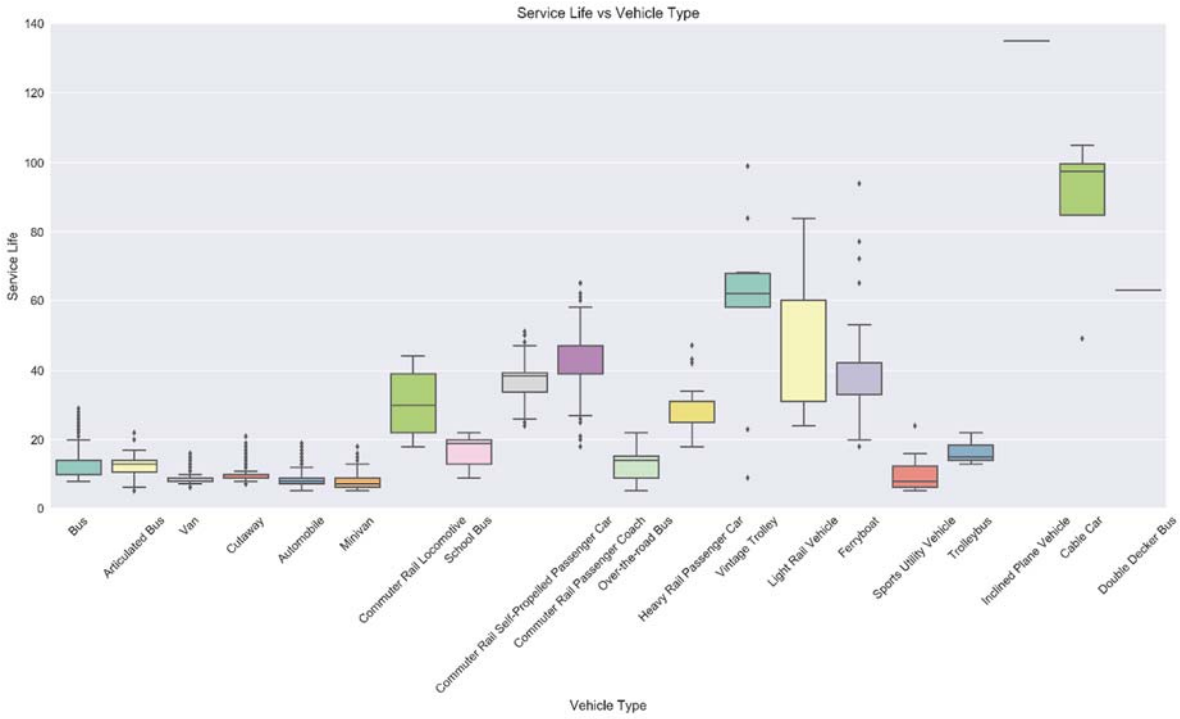


Figure 15. Box Plot of Service Life by Vehicle Type

The *corrcoef()* function was applied to the six feature columns that was visualized in the scatter plot matrix. Then, the *heatmap()* function was applied to the correlation matrix that was plotted as a heat map. The code block is as follows:

```
# Service Life correlation matrix
k = 10
# Matrix form for correlation data
corrmat = df_analysis.corr()
cols = corrmat.nlargest(k, 'Service_Life')['Service_Life'].index
cm = np.corrcoef(df_analysis[cols].values.T)
sns.set(font_scale = 1.25)
f, ax = plt.subplots(figsize = (11, 9))
hm = sns.heatmap(cm, cbar = True, annot = True, square = True, fmt =
      '.2f', annot_kws = {'size': 12}, yticklabels = cols.values,
      xticklabels = cols.values, linecolor = 'white', linewidths = 1)
save_image('heatmap')
```

The above code generated the correlation matrix provides a summary with graphic representation as shown in Figure 16. This graphic summary was analyzed for features correlations (Mirjalili & Raschka, 2017). This heat map indicates that the target variable Service Life does not have a strong correlation with any of the features; the strongest correlation is 0.42 for the feature Seating Capacity. By analyzing the scatter plot and correlation matrix, a non-linear relationship between target variable and other features was found. Therefore, the linear regression model was not a good choice for this problem. Thus, a non-linear regression model was applied.

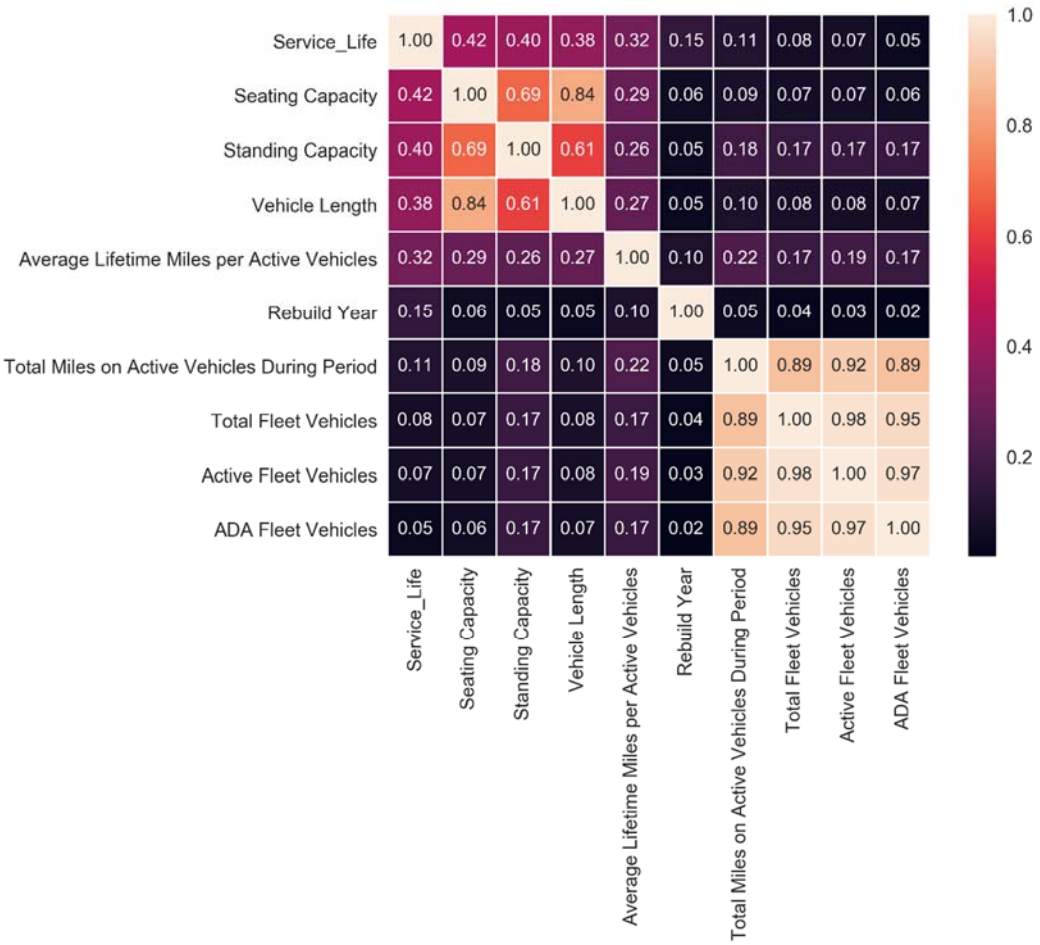


Figure 16. Heat map of Correlation Matrix with Features

4.7. Preprocessing the Training Data

Features of the training data set needed to be engineered before building a machine learning model. In order to engineer the training data, the previously preprocessed training data were loaded and stored into a data frame by executing the following code:

```
# Read cleaned training data
df = pd.read_csv('..//NTD/initial_training.csv')
```

4.7.1. Create new features

The feature engineering process involves determining which features need to be used, what iterative processes need to be required for feature selection, and what combination of features need to be added for making predictions (Downey, 2014). In this problem, eight new features were created by combining different numerical features. The following sample block of code created the feature StandingCap_SeatingCap dividing 'Standing Capacity' by 'Seating Capacity'. The code also replaced null values with zero in the column.

```
# Create new feature
df['StandingCap_SeatingCap'] = df['Standing Capacity']/df['Seating
      Capacity']
df['StandingCap_SeatingCap'].replace(np.inf, 0, inplace = True)
```

Similarly, the other new features VehicleLength_SeatingCapacity, VehicleLength_StandingCapacity, TMOAVDP_TFV, TMOAVDP_AFV, ALMPAV_TFV, ALMPAV_AFV, and RebuildYear_ManufactureYear, were created following the same pattern in which the first variable (before underscore) was divided by the second variable (after underscore).

4.7.2. Create additional features from categorical features

Scikit-learn supports binary encoding by using the LabelBinarizer class that is available in the Scikit-learn's preprocessing package. It converts multiple labels to binary labels. The *fit()* method picks the parameters from the data and the *transform()* method applies the parameters to the new data (Massaron & Boschetti, 2016). The LabelBinarizer method was applied on Fuel Type, Vehicle Type, Funding Source, Mode, and Ownership Type. The new features using LabelBinarizer were renamed by prefixing the category name. The block of codes on Fuel Type using LabelBinarizer is shown below. Codes on other types were written in the similar manner.

```
# Import class
from sklearn import preprocessing
# Binarize columns
lb = preprocessing.LabelBinarizer(pos_label = 1, neg_label = 0,
      sparse_output = False)
# Fit label binarizer
lb.fit(['Compressed Natural Gas', 'Diesel Fuel', 'Diesel
      Fuel/Compressed Natural Gas', 'Diesel Fuel/Electric Propulsion
      Power', 'Electric Battery', 'Electric Propulsion Power',
      'Ethanol', 'Gasoline', 'Gasoline/Liquefied Petroleum Gas',
      'Hybrid Diesel', 'Hybrid Gasoline', 'Hydrogen Cell', 'Liquefied
      Natural Gas', 'Liquefied Petroleum Gas', 'Other', 'Unknown
      Fuel'])
# Join the categorical features with the numerical features
df = df.join(pd.DataFrame(data = lb.transform(df['Fuel Type']), columns
      = [lb.classes_]).applymap(func = bool))
# Rename binarized columns
df.rename(columns = {'Compressed Natural Gas': 'Fuel Type_Compressed
      Natural Gas', 'Diesel Fuel': 'Fuel Type_Diesel Fuel', 'Diesel
      Fuel/Compressed Natural Gas': 'Fuel Type_Diesel Fuel/Compressed
      Natural Gas', 'Diesel Fuel/Electric Propulsion Power': 'Fuel
      Type_Diesel Fuel/Electric Propulsion Power', 'Electric Battery':
      'Fuel Type_Electric Battery', 'Electric Propulsion Power': 'Fuel
```

```
Type_Electric Propulsion Power', 'Ethanol': 'Fuel Type_Ethanol',
'Gasoline': 'Fuel Type_Gasoline', 'Gasoline/Liquefied Petroleum
Gas': 'Fuel Type_Gasoline/Liquefied Petroleum Gas', 'Hybrid
Diesel': 'Fuel Type_Hybrid Diesel', 'Hybrid Gasoline': 'Fuel
Type_Hybrid Gasoline', 'Hydrogen Cell': 'Fuel Type_Hydrogen
Cell', 'Liquefied Natural Gas': 'Fuel Type_Liquefied Natural
Gas', 'Liquefied Petroleum Gas': 'Fuel Type_Liquefied Petroleum
Gas', 'Other': 'Fuel Type_Other', 'Unknown Fuel': 'Fuel
Type_Unknown Fuel'}, inplace = True)
```

4.7.3. Create features with dummy variables

A convenient way to create dummy features for machine learning applications is to transform a categorical variable into a dummy matrix. If a string column in a data frame has *n* values, the *get_dummies()* function will convert *n* columns into 1's or 0's (McKinney, 2017). In this training data, the categorical string columns TOS and Dedicated Fleet were converted into dummy variables using the *get_dummies()* function. The code is as follows:

```
# Replace categorical data with one-hot encoded data
df = pd.get_dummies(data = df, columns = ['TOS', 'Dedicated Fleet'])
```

4.7.4. Create features by analyzing the histogram of various categorical features

Histograms categorize data into bins. Although each bin contains a default data range of 10, the data range can be set by the user. Histogram plots the items in each bin and the distribution of data can be visualized from bin to a bin (Mueller & Massaron, 2015). Five additional features were created through analyzing histograms on Service Life against five categorical features called Fuel Type, Vehicle Type, Mode, Funding Source, and Ownership Type. Values for the newly created features were chosen based on the patterns of the histograms

and the mean values of Service Life in each category. The following code plotted histograms on Service Life by Fuel Type.

```
# plot histograms
df_stats.loc[:, ['Fuel_Type',
      'Service_Life']].groupby('Fuel_Type').hist()
```

The above code generated a series of histograms. Due to space constraint, only the histogram for Service Life by Compressed Natural Gas is included in Figure 17. The below histogram showed that the service life of most of the vehicles fell between 14 and 15 years.

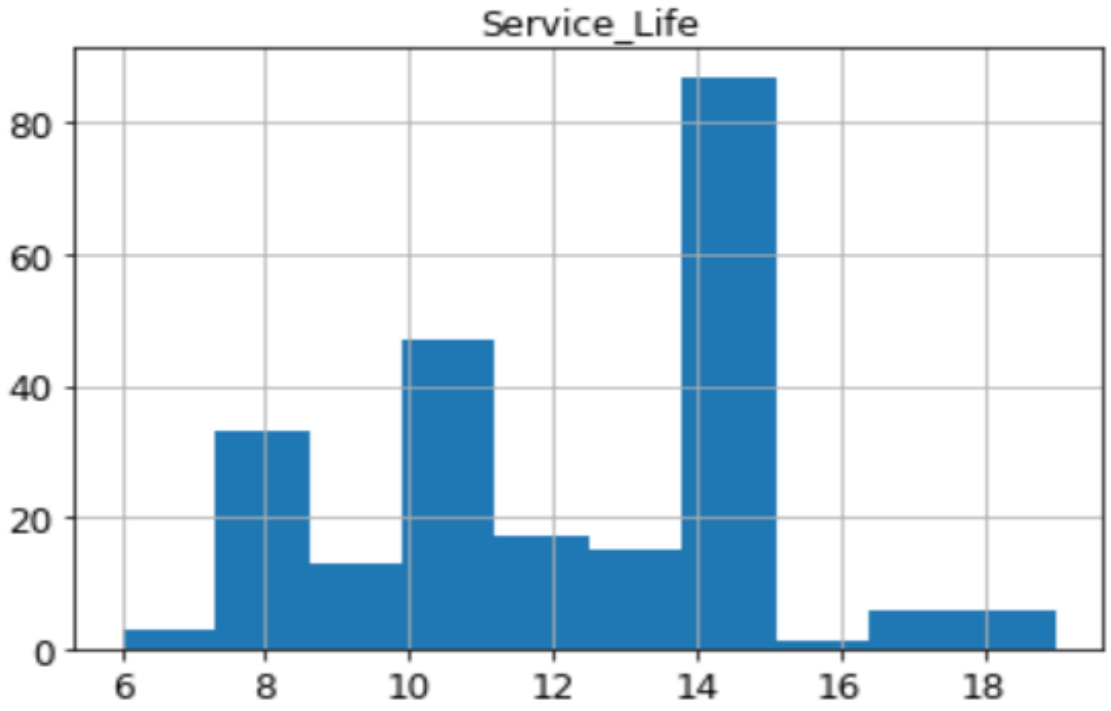


Figure 17. Histogram of Service Life vs. Number of Vehicles with Compressed Natural Gas

The following code calculated the average service life, the maximum service life, and the standard deviation of vehicles in each fuel category.

```
# Calculate mean, max and standard dev. of Service Life by Fuel Type
df_stats.groupby('Fuel_Type').Service_Life.agg(['max', 'mean', 'std'])
```

The mean, the max, and the standard deviation of service life by Fuel Type are shown in Table 18. The average service life of vehicles in the compressed natural gas category was 12 years, the maximum service life was 19 years, and a standard deviation of about three years. Therefore, after visualizing the above histogram as well as the statistical analysis of service life, the Service Life by Compressed Natural Gas was mapped by 15 years. Similarly, all other service life was calculated and mapped accordingly.

Table 18. Statistical Analysis of Service Life by Fuel Type

Fuel Type	Max	Mean	STD
Compressed Natural Gas	19	12.041	2.744
Diesel Fuel	94	12.943	5.680
Diesel Fuel/Compressed Natural Gas	39	13.671	5.478
Diesel Fuel/Electric Propulsion Power	11	11	-
Dual Fuel	15	11.5	4.949
Electric Battery	22	14.733	4.131
Electric Propulsion Power	135	38.5	20.246
Ethanol	14	14	0
Gasoline	42	9.608	3.018
Gasoline/Liquefied Petroleum Gas	14	10.5	4.041
Hybrid Diesel	14	11.093	3.165
Hybrid Gasoline	14	12.583	2.574
Hydrogen Cell	14	14	0
Liquefied Natural Gas	21	12.312	2.889
Liquefied Petroleum Gas	14	10.904	2.527
Other	14	10.75	2.217
Unknown Fuel	65	12.985	9.881

The following block of code showed the mapping of service life by fuel type:

```
# Map fuel type with the service life
df['Fuel Type_Service Life'] = df['Fuel Type'].map({'Compressed Natural
      Gas': 15, 'Diesel Fuel': 16, 'Diesel Fuel/Compressed Natural
      Gas': 17, 'Diesel Fuel/Electric Propulsion Power': 11, 'Electric
      Battery': 19, 'Electric Propulsion Power': 56, 'Ethanol': 14,
      'Gasoline': 10, 'Gasoline/Liquefied Petroleum Gas': 7, 'Hybrid
      Diesel': 10, 'Hybrid Gasoline': 7, 'Hydrogen Cell': 3, 'Liquefied
      Natural Gas': 15, 'Liquefied Petroleum Gas': 12, 'Other': 10,
      'Unknown Fuel': 32})
```

Similarly, four other additional features, Service Life_Vehicle Type, Service Life_Funding Source, Service Life_Mode, and Service Life_Ownership Type, were created by mapping service life values.

4.7.5. Remove unnecessary columns

Since the creation of new features was done with the Manufacture Year column and other categorical feature columns, those columns were no longer needed and removed from the training data set by executing the following code:

```
# Remove the unnecessary fields from the data set
df.drop(['Manufacture Year', 'Fuel Type', 'Vehicle Type', 'Funding
      Source', 'Mode', 'Supports Mode', 'Ownership Type'], axis =
      'columns', inplace = True)
```

4.7.6. Check null values in the training data

The following code checked whether there were any null values in the training data set:

```
# Checking Null values in the data set
df.isnull().sum()
```

The output is listed in Table 19 (only 10 of the features out of 120 are shown). In the output window, the number in the right column of each feature indicated how many null values existed in the data. If any null values existed, the data set needed to be fixed by removing null values; otherwise, it would fail to build a model using the machine learning algorithm. The value 0 (zero) indicated the data set was ready for training the predictive model.

Table 19. Null Values in the Data Set

Features Name	Number of Null Points
Revenue Vehicle Inventory ID	0
ADA Fleet Vehicles	0
Active Fleet Vehicles	0
Average Lifetime Miles per Active Vehicles	0
Emergency Contingency Vehicles	0
Rebuild Year	0
Seating Capacity	0
Standing Capacity	0
Total Fleet Vehicles	0
Total Miles on Active Vehicles During Period	0

4.7.7. Set index

The following code was used to set the index of the training data as the Revenue Vehicle Inventory ID field:

```
# Set index to Revenue Vehicle Inventory ID
df = df.set_index('Revenue Vehicle Inventory ID')
```

4.7.8. Check the number of rows and columns in training data set

The following code was used to check the number of data points and features in the training set to train the predictive model:

```
# Checking the number of rows and columns in the training data set
df.shape
```

The output showed a tuple of (7745, 119), which meant there were 7745 rows with 119 columns in the training data set.

4.7.9. Save the training data

Finally, the following code was used to save the training data in the training.csv file in the same directory in the iPython Notebook.

```
# Save the training data
df.to_csv('training.csv', sep = ',')
```

4.8. Create Deployment Data Set for Prediction

The revenue vehicle deployment data set consisted of data of vehicles in operation. After building the model with the training data set, the model was applied to the deployment data set to predict the service life of vehicles. There were 31149 data points in the deployment data set, which indicated 31149 vehicles were in operation nationwide based on revenue vehicle data from 2008 to 2016. The deployment data were separated from total vehicles from 2008 to 2016 based on the N flag in the Retired column. The main purpose of creating the deployment data set was to predict the service life of vehicles still in operation. Since the machine learning method works only when the X features in the training data set match the X features in the deployment data set exactly, the processing of the deployment data set was done in the same way as processing the training data set was done. Finally, the processed deployment data set was saved in a CSV file by executing the below code:

```
# Save data to .csv file
X_deploy.to_csv('Final Deployment Data.csv', sep = ',')
```

4.9. Develop Simple Linear Regression Model using SAS

A simple linear regression model was developed using the Statistical Analysis System (SAS) software to see whether the predictive model could be useful for this problem. The simple linear regression model using the full training set with the top 23 important features produced the performance results shown in Table 20. The value of R^2 in the full training set is 0.7184, which indicates that the model explains 72% of the variance in the data set. The RMSE score of 3.77945 indicates the prediction falls within 3.78 years below or above the standard deviation with a 72% accuracy. Therefore, the below results indicate that the simple linear regression was not a good fit for this problem; thus it was not considered as a viable model for this problem.

Table 20. Performance Measures with Simple Linear Regression by SAS

Performance Measures	Performance Scores
Root Mean Squared Error (RMSE)	3.77945
Dependent Mean	12.10471
The Coefficient of Variation (Coeff Var)	31.22299
R^2 Score	0.7184
Adjusted R^2 Score	0.7175

4.10. Develop Predictive Model

After outlining the initial parameters for each module (as detailed above), the training.csv file was loaded into a data frame called training. The code is as follows:

```
# Load training data
training = pd.read_csv('..//NTD/training.csv')
```

After loading, the training data was split to separate the target variable Service Life from predictor variables. The following block of code loaded the predictor variables into an object called *X*, and the Service Life variable into an object called *y*:

```
# Create the X arrays
X = training.set_index('Revenue Vehicle Inventory ID')
# Create the y arrays
y = X.pop('Service Life')
```

The shape attribute checks the number of rows and columns in the data frame *X* and the *y* series.

```
# Check the shape of the X features
X.shape
# Check the shape of the y response
y.shape
```

The above codes showed 7745 rows and 118 columns in the *X* data frame and 7745 rows in the *y* series in a single column.

4.10.1. Random forest regression model

The random forest regression is an ensemble technique that combines multiple decision trees. Because it can randomize, the random forest regression handles generalization better than an individual decision tree; thus, the variance of the model decreases (Mirjalili & Raschka, 2017). Before building the predictive model with the training data, the hyperparameters for RandomForestRegressor class were tuned to train a random forest model.

4.10.1.1. Tuning hyperparameters for random forest regression model

The hyperparameters, n_estimators, max_features, and min_sample_leaf, were tuned to the training data to increase the predictive performance. The default value of the n_estimators was 10; this default value needed to be tuned for the best results. Therefore, a series of the number of trees were selected to find the best value for n_estimators. The following block of code assigned a series of number of estimators to ascertain which value returned the best root mean squared error (RMSE) on the training data set:

```
# Empty tree list
tree_results = []
n_estimator_options = [25, 50, 75 ,100, 125, 150, 175, 200, 300, 400,
      500, 600, 700, 800, 900, 1000]
for trees in n_estimator_options:
      model = RandomForestRegressor(trees, oob_score = True, n_jobs = -
      1, random_state = 42)
model.fit(X, y)
rmse = np.sqrt(mean_squared_error(y, model.predict(X)))
tree_results.append(rmse)
```

The above results were made into a graph using the following code:

```
# Set plot style
plt.style.use('ggplot')
colors = ['lightcoral' if c == min(tree_results) else 'cornflowerblue'
      for c in tree_results]
```

```
ax = pd.Series(tree_results, n_estimator_options).plot(kind = 'barh',
      color = colors, xlim = [min(tree_results)-0.5, max(tree_results)
      + 0.5], figsize = (12,5) )
ax.set_ylabel('Number of Trees')
ax.set_xlabel('Root Mean Squared Error')
```

The above code generated the graph shown in Figure 18 showed that the lowest RMSE value was achieved while the number of trees (n_estimators) was 500.

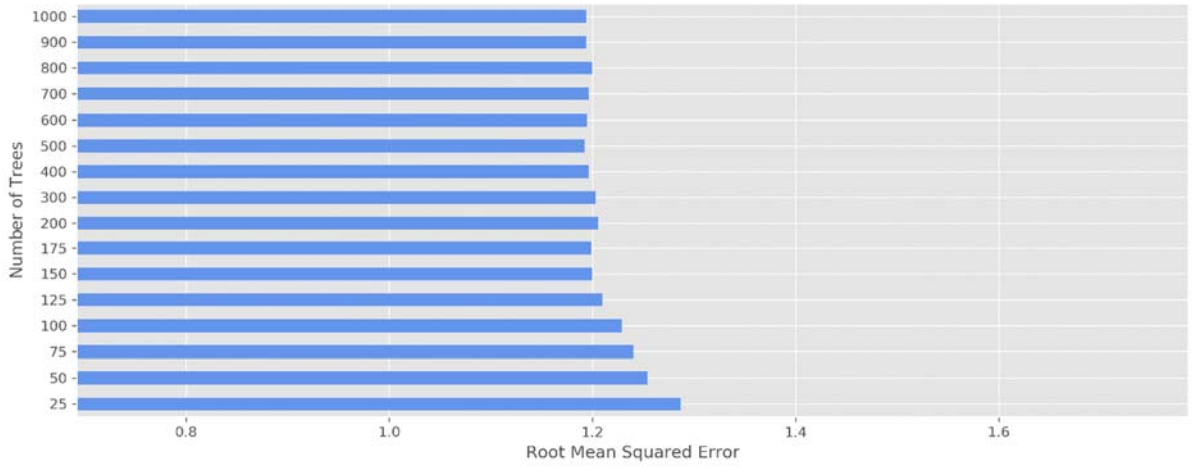


Figure 18. A Bar Plot of Number of Trees vs. Root Mean Squared Error

The parameter max_features needed to be optimized because it originally defaulted to 'None'. The number of features to be considered was based on the number features in the training data and the problem. The maximum features were preconfigured with parameter options such as 'auto' for all features, 'sqrt' or 'log' functions on the number of features, as well as the percent of all features. The following block of code was set to produce the best max_features parameter making sure to set n_estimators with 500:

```
# Empty list for max_features
max_features_results = []
max_feature_options = ['auto', None, 'sqrt', 'log2', 0.9, 0.8, 0.7,
      0.6, 0.5, 0.4, 0.3, 0.2, 0.1]
for max_features in max_feature_options:
```

```
        model = RandomForestRegressor(n_estimators = 500, oob_score =
        True, n_jobs = -1, random_state = 42, max_features =
        max_features)
    model.fit(X, y)
    rmse = np.sqrt(mean_squared_error(y, model.predict(X)))
    max_features_results.append(rmse)
    # Set plot style
    plt.style.use('ggplot')
    colors = ['lightcoral' if c == min(max_features_results) else
        'cornflowerblue' for c in max_features_results]
    ax = pd.Series(max_features_results, max_feature_options).plot(kind =
        'barh', color = colors, xlim = [min(max_features_results) - 0.5,
        max(max_features_results) + 0.5], figsize = (12, 5));
    ax.set_ylabel('max_features')
    ax.set_xlabel('Root Mean Squared Error')
    save_image('max_features_rmse')
```

The above block of code generated the plot shown in Figure 19 that shows that the model produced the best result when the maximum number of features was set to 70% of all features (max_features = 0.7).

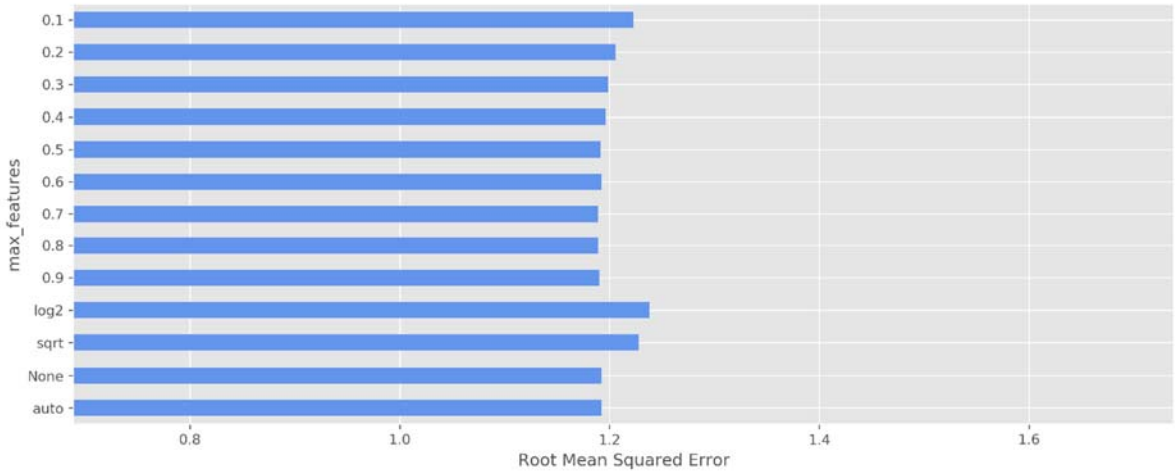


Figure 19. A Bar Plot of Maximum Features vs. Root Mean Squared Error

The min_samples_leaf parameter was run by setting max_features = 0.7 and n_estimators = 500. The default value for this parameter is 1, which is good for a first few training-runs on the

data set. Assigning a series of values from 1 to 10 for this parameter and running the below code will produce the best performance.

```
# Create empty sample leap
min_sample_leaf_results = []
min_sample_leaf_options = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
for min_samples in min_sample_leaf_options:
      model = RandomForestRegressor(n_estimators = 500, oob_score =
            True, n_jobs = -1, random_state = 42, max_features = 0.7,
            min_samples_leaf = min_samples)
model.fit(X, y)
rmse = np.sqrt(mean_squared_error(y, model.predict(X)))
min_sample_leaf_results.append(rmse)
# Set pandas series
ax = pd.Series(min_sample_leaf_results,
      min_sample_leaf_options).plot(figsize = (12, 5), color =
      'cornflowerblue');
ax.set_xlabel('min sample leaf')
ax.set_ylabel('Root Mean Squared Error')
save_image('sample_leaf_rmse')
```

The above code plotted the graph below in Figure 20 showing that the default value (1) of min sample leaf produced the best result.

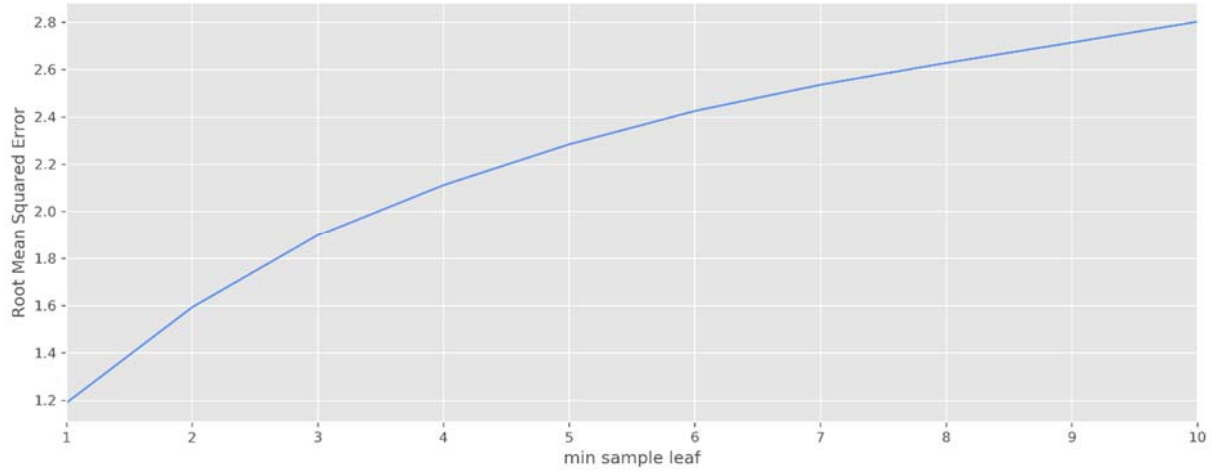


Figure 20. A line Plot of min_sample_leaf vs. Root Mean Squared Error (RMSE)

After tuning the hyperparameters for the random forest regression predictive model, the following hyperparameters were selected to optimize the state of good repair predictive model.

```
# Parameters for RFR
RandomForestRegressor(n_estimators = 500, oob_score = True, n_jobs = -
      1, random_state = 42, max_features = 0.7, min_samples_leaf = 1)
```

4.10.1.2. Building a random forest model to predict the service life of vehicles

The machine learning model was built using the Scikit-learn four-step modeling pattern. In step one, the random forest regression class was imported. In step two, the model was instantiated with the estimator by setting hyper-parameters that were tuned earlier. The tuned parameters were instantiated by setting the max_features to 0.7, the n_estimators to 500, and the min_samples_leaf to 1 in the RandomForestRegressor object. In step three, the model was fit on the training data and then the patterns that were learned from the data were stored in the memory. In step four, the fitted model was applied to predict the response variable to the test set for evaluation (Inyang, Ozuomba, & Ezenkwu, 2017).

Before building any predictive model, it is important to test the model on unseen data to evaluate its performance. Therefore, first the training data were split into the train set and the test set; the model was fit to the train set and evaluated on the test set (Raschka, 2015). The following code showed the *train_test_split()* function that was used to split the training data into the train set, 70% of the data, and the test set, 30% of the data.

```
# Import the class
from sklearn.model_selection import train_test_split
# Split the data set in a training set (2/3) and a test set (1/3)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state
      = 42, test_size = 0.30)
```

The shape attribute provided the number of rows and number of columns in a tuple. The following block of codes showed the number of rows that were split into the train set and the test set by running shape attribute in the data frame:

```
X_train.shape
```
Output: (5421, 118)

```
y_train.shape
```
Output: (5421L,)

```
X_test.shape
```
Output: (2324, 118)

```
y_test.shape
```
Output: (2324L,)

The above output showed that 5421 data points were allotted for training and 2324 data points for testing the model.

First, the RandomForestRegressor object with tuned hyperparameters was instantiated, and then the *fit()* method was applied on the X_train and y_train sets. After that, the *predict()* method was invoked on the X_test set, which then generated predictions (Mirjalili & Raschka, 2017). In addition, the *predict()* method was invoked on X_train set for comparing performance measures with the test set. The block of codes is as follows:

```
# Instantiate Random Forest Regressor with tuned hyperparameters
rfr_eval = RandomForestRegressor(n_jobs = -1, n_estimators = 500,
      oob_score = True, random_state = 42, max_features = 0.7,
      min_samples_leaf = 1)
# Fit the model to the training data
rfr_eval.fit(X_train, y_train)
# Make the predictions on the train set
y_pred_train = rfr_eval.predict(X_train)
# Make the predictions on the test set
y_pred_test = rfr_eval.predict(X_test)
```

After having fitted the model with the training data, the model was evaluated on the test set as well as on the train set by applying the performance measures of RMSE, MAE, and R^2 score to see how well the model worked on the unseen data. If the performance results were satisfactory for generalization errors, the model could be used to predict future data. If the performance results are not acceptable, the model needed to be tuned further for optimal performance (Mirjalili & Raschka, 2017). The block of codes on the train set with results is as follows:

```
# Find the error rate on the train set
rms_train = np.sqrt(mean_squared_error(y_train, y_pred_train))
mae_train = mean_absolute_error(y_train, y_pred_train)
r2_train = r2_score(y_train,y_pred_train)
print('Root Mean Squared Error:\t\t%0.2f' % rms_train)
print('Mean Absolute Error:\t\t\t%0.2f' % mae_train)
print('R2 Score:\t\t%0.2f' % r2_train)
```

The performance results of random forest regression model on the train set are listed below in Table 21.

Table 21. The Performance Measures with Random Forest Regression on Training Set

Performance Measures	Performance Scores
Root Mean Squared Error (RMSE)	1.27
Root Mean Squared Error (MAE)	0.72
R^2 Score	0.97

The block of codes on the test set with performance results is as follows:

```
# Find the error rate on test data
rms_test = np.sqrt(mean_squared_error(y_test, y_pred_test))
mae_test = mean_absolute_error(y_test, y_pred_test)
r2_test = r2_score(y_test,y_pred_test)
print('Root Mean Squared Error:\t\t%0.2f' % rms_test)
print('Mean Absolute Error:\t\t\t%0.2f' % mae_test)
print('R2 Score:\t\t%0.2f' % r2_test)
```

The performance results of the predictive model on the test set are listed below in Table 22.

Table 22. The Performance Measures with Random Forest Regression on Test Set

Performance Measures	Performance Scores
Root Mean Squared Error (RMSE)	3.39
Root Mean Squared Error (MAE)	1.94
R^2 Score	0.78

The performance results in Table 21 and Table 22 were shown in Table 23 side by side and compare the results on the train set to the test set. The comparison results between the train set and the test set showed that the RMSE value of 3.39 on the test set was much larger than the RMSE value of 1.27 on the train set. This difference was an indicator that the current model was overfitting the train data. In machine learning problems, overfitting is common when the model performs well on the train data but does not generalize well on the test or unseen data. The model may have a high variance due to overfitting. In addition, many parameters in the model may cause the model to be too complex. Therefore, the noise can be filtered out from the data by tuning parameters and removing non-important features from the model (Mirjalili & Raschka, 2017).

Table 23. Comparison of Performance Results on the Training Set and the Test Set using the Random Forest Regression Method

Method	Train Set			Hold-Out Set (Test Set)		
	RMSE	MAE	R^2 Score	RMSE	MAE	R^2 Score
RFR	1.27	0.72	0.97	3.39	1.94	0.78

4.10.1.3 Building a random forest model with full data set as the training set

The following block of code illustrates the 4 steps random forest regression model:

```
# Import the class
from sklearn.ensemble import RandomForestRegressor
# Instantiate Random Forest Regressor
```

```
rfr = RandomForestRegressor(n_jobs = -1, n_estimators = 500, oob_score =
      True, random_state = 42, max_features = 0.7, min_samples_leaf =
      1)
# Fit regression model
rfr.fit(X,y)
#Make the predictions on the training set
y_pred = rfr.predict(X)
```

After building a machine learning model, it needs to be measured for performance. The following block of code was used for performance measure:

```
# Find the error rate on the full set of training data
rmse = np.sqrt(mean_squared_error(y, y_pred))
mae = mean_absolute_error(y, y_pred)
r2 = r2_score(y, y_pred)
print('Root Mean Squared Error:\t\t%0.2f' % rmse)
print('Mean Absolute Error:\t\t\t%0.2f' % mae)
print('R2 Score:\t\t%0.2f'% r2)
```

The performance results of the predictive model on the full data set are listed in Table 24. The RMSE result of 1.23 in the random forest regression model performed well because the prediction error is up to 1 year or above. The MAE of 0.71 is acceptable. The value of R^2 in the train set is 0.97, which indicates that the model explains 97% of the variance in the training set. The below results suggested a good result, but not the best result. Since the evaluation results of the random forest regression model do not seem to generalize well enough to deploy for prediction, few more regression algorithms were applied before choosing the best model.

Table 24. Performance Measures with Random Forest Regression on Full Training Set

Performance Measures	Performance Scores
Root Mean Squared Error (RMSE)	1.23
Root Mean Squared Error (MAE)	0.71
R^2 Score	0.97

4.10.2. Gradient boosting regression model

Tree-based ensemble methods combine simple regression trees with poor results, fit complex non-linear relationships, and produce high-performance predictions. The gradient boosting regression method corrects the prediction made by previous base models in order to improve prediction accuracy. In this problem, the gradient boosting regression tree method was applied to build the model for service life on revenue vehicle inventory data in order to improve prediction accuracy as compared to the random forest regression model (Zhang & Haghani, 2015).

4.10.2.1. Tuning hyperparameters for gradient boosting regression model

In scikit-learn, hyperparameters are parameters that are passed as arguments to the constructor of the classes (Pedregosa, et al., 2011). The gradient boosting regression has many parameters that can be tuned, such as learning_rate, n_features, max_features, min_samples_split, max_depth, and min_samples_leaf. Before tuning hyperparameters using GridSearchCV, a few required scikit-learn's libraries were imported. The following block of code loaded the training data and created *X* variables and *y* variable:

```
# Import classes
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
# Import data science package
import pandas as pd
# Load training data
training = pd.read_csv('..//NTD/training.csv')
# Create the X arrays
X = training.set_index('Revenue Vehicle Inventory ID')
# Create the y arrays
y = X.pop('Service Life')
```

The *train_test_split()* utility function was applied to split the data into a train set and a test set. The train set was fed to the GridSearchCV instance and the test set was used to compute performance metrics. The GridSearchCV instance provided a grid search in the parameters of 'param_grid' and generated the best parameters from a grid of parameters (Pedregosa, et al., 2011). The parameters were set as follows:

```
# Set a dictionary of parameters
param_grid = {'learning_rate': [0.1, 0.01, 0.001], 'max_depth': [2, 4,
      6, 8, 10], 'min_samples_leaf': [2, 3, 4, 5, 6],
      'min_samples_split': [2, 3, 4, 5, 6], 'max_features': [1.0, 0.8,
      0.7, 0.6, 0.5]}
```

The parameter grid was set with a wide range of parameters for the grid search. The learning_rate parameter controls the output of each tree and determines how fast or how slow it can converge to the optimal result. The max_depth defines maximum depth of a tree controls overfitting and allows the model to learn relations. The min_samples_leaf parameter defines minimum samples in a leaf and it also controls overfitting. The max_features defines the number of features and the min_samples_split parameter defines the minimum number of observations that will be considered for splitting (Jain, 2016).

Now, after instantiating the GradientBoostingRegressor model with 3000 trees, the model was fit with the training set and was run with GridSearchCV instance. Since a wide range of parameters and a higher number of trees were used in the grid search, the iterations took some time to finish. The block of codes was as follows:

```
# Instantiate the model
est = GradientBoostingRegressor(n_estimators = 3000)
# Grid Search
gs_cv = GridSearchCV(est, param_grid, scoring = 'mean_squared_error',
      n_jobs = -1).fit(X_train, y_train)
```

After finishing up the grid search iterations, the following code found the best parameters:

```
# Get the best hyperparameters
print('Best hyperparameters: %r' % gs_cv.best_params_)
```

The above code generated the following output:

```
Best hyperparameters: {'max_features': 0.8, 'min_samples_split': 5,
'learning_rate': 0.01, 'max_depth': 10, 'min_samples_leaf': 5}
```

After generating all the hyperparameters, the gradient boosting regression model was ready to build the predictive model with the revenue vehicle inventory training data to solve transit state of good repair issues.

4.10.2.2. Building and evaluating a gradient boosting regression predictive model

The gradient boosting regression model needed to be tested on unseen data set to ascertain its performance. This is because the model, even if all of revenue vehicle data are used and estimate the performance on the same data, the model may not provide an accurate picture of its performance on unseen data. For this reason, it is important to split the data into the train set and the test set, and then train the model with the train set by setting best hyperparameters. The following block of code provided the evaluation procedure on test data:

```
# Import class
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split
# Split the data set in a training set (2/3) and a test set (1/3)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state
      = 42, test_size = 0.30)
# Instantiate regression model with tuned hyperparameters
gbr_eval = GradientBoostingRegressor(n_estimators = 3000, max_features
      = 1.0, min_samples_split = 5, learning_rate = 0.01, max_depth =
      6, min_samples_leaf = 5, loss = 'ls')
# Fit the model
gbr_eval.fit(X_train,y_train)
```

```
# Make the predictions on the train set
y_pred_train = gbr_eval.predict(X_train)
# Make the predictions on the test set
y_pred_test = gbr_eval.predict(X_test)
```

The following block of code calculated the performance measures by gradient boosting regression model on the train set:

```
# Find the error rate on the train data set
rms_train = np.sqrt(mean_squared_error(y_train, y_pred_train))
mae_train = mean_absolute_error(y_train, y_pred_train)
r2_train = r2_score(y_train,y_pred_train)
print('Root Mean Squared Error:\t\t%0.2f' % rms_train)
print('Mean Absolute Error:\t\t\t%0.2f' % mae_train)
print('R2 Score:\t\t%0.2f' % r2_train)
```

The performance results of the predictive model on the train set are shown in Table 25.

Table 25. The Performance Measures with Gradient Boosting Regression on Training Set

Performance Measures	Performance Scores
Root Mean Squared Error (RMSE)	1.52
Root Mean Squared Error (MAE)	1.05
R^2 Score	0.95

The following block of code on the test set calculates performance measure:

```
# Find the error rate on test data set
rms_test = np.sqrt(mean_squared_error(y_test, y_pred_test))
mae_test = mean_absolute_error(y_test, y_pred_test)
r2_test = r2_score(y_test,y_pred_test)
print('Root Mean Squared Error:\t\t%0.2f'% rms_test)
print('Mean Absolute Error:\t\t\t%0.2f'% mae_test)
print('R2 Score:\t\t%0.2f'% r2_test)
```

The performance results of the predictive model on the train set are shown in Table 26.

Table 26. The Performance Measures with Gradient Boosting Regression on Test Set

Performance Measures	Performance Scores
Root Mean Squared Error (RMSE)	1.37
Root Mean Squared Error (MAE)	0.94
R^2 Score	0.96

The performance results on the train set and the test set are shown below in Table 27 side by side and compared to the performance results. This table shows that the RMSE score on the train set is very close to RMSE score on the test set. Based on these figures, there is no indication of overfitting in the model; thus this table generalizes the model very well. Therefore, this model can be used for predictions. Furthermore, the full training set can be used to train the model; this will further improve the performance because the full training data set contains more training data.

Table 27. Comparison of Performance Results on Training Set and Test Set with Gradient Boosting Regression Method

Method	Train Set			Hold-Out Set (Test Set)		
	RMSE	MAE	R^2 Score	RMSE	MAE	R^2 Score
GBR	1.52	1.05	0.95	1.37	0.94	0.96

4.10.2.3. Building a gradient boosting regression model with full data as the training set

Since the performance results indicate a good predictive model during evaluation, the training set was not split further for evaluation. Instead, the full training set was used to train the model before applying to the deployment set. Just like random forest regression model, the *fit()* and *predict()* methods in scikit-learn were used in the same way to build the model. The following block of codes was used to develop the gradient boosting regression model:

```
# Import the class
from sklearn.ensemble import GradientBoostingRegressor
# Instantiate regression model with tuned hyperparameters using least-
# squares
```

```
gbr = GradientBoostingRegressor(n_estimators = 3000, max_features =
      0.8, min_samples_split = 5, learning_rate = 0.01, max_depth = 10,
      min_samples_leaf = 5, loss = 'ls')
# Fit the Gradient Boosting Regression model
gbr.fit(X, y)
# Make predictions on the overall data set
y_gbr_pred = gbr.predict(X)
```

The following block of code calculated the performance measures by the gradient boosting regression model on the overall data set:

```
# Find the error rate on the full set
rmse = np.sqrt(mean_squared_error(y, y_gbr_pred))
mae = mean_absolute_error(y, y_gbr_pred)
r2 = r2_score(y, y_gbr_pred)
print('Root Mean Squared Error:\t\t%0.2f' % rmse)
print('Mean Absolute Error:\t\t\t%0.2f' % mae)
print('R2 Score (Variance Score):\t\t%0.2f' % r2)
```

The performance results of the predictive model on full data set are shown in Table 28. The performance results indicate a very good performance model with the gradient boosting regression. The R^2 score suggested that the model can predict with a 98% accuracy about 1.04 years above or below the mean year, with a minimum absolute error of 0.65. Therefore, the gradient boosting regression predictive model was a good fit for this problem of predicting the service life of transit vehicles.

Table 28. The Performance Measures with Gradient Boosting Regression on Full Data Set

Performance Measures	Performance Scores
Root Mean Squared Error (RMSE)	1.04
Root Mean Squared Error (MAE)	0.65
R^2 Score	0.98

4.10.3. Decision tree regression predictive model

Nevertheless, in order to fully explore this problem, another algorithm, the decision tree regression, was applied and the results were analyzed before choosing the best model for the problem. A decision tree builds a regression model in the form of a tree-like structure to solve regression problems and is a good fit to handle the complex nonlinear relationship between features variables and target variable. A decision tree is a top-down approach where the processing breaks down a data set into smaller subsets while at the same time the tree moves down until the leaf node. The basic idea is to break down complex decisions into smaller subsets of simpler decisions so that it is easier to arrive at a solution. In a regression problem, the decision tree considers features of data as predictor variables and the continuous variable as the target variable. The features with important information are chosen for the model, and features with no information are rejected automatically from the model, thus increasing the computational efficiency (Xu, Watanachaturaporn, Varshney, & Arora, 2005).

4.10.3.1. Tuning hyperparameters for decision tree regression model

A grid search algorithm was applied to find the optimal hyperparameters for the decision tree regression model. The following block of codes imported some required classes and data science packages for tuning hyperparameters:

```
# Import classes
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
# Import Data Science package
import pandas as pd
```

The training data set was loaded and split into the train set and the test set. The following block of code performed the grid search on the train set only:

```
# Load the training data
training = pd.read_csv('..//NTD/training.csv')
# Create the X arrays
X = training.set_index('Revenue Vehicle Inventory ID')
# Create the y arrays
y = X.pop('Service Life')
# Split the training data into a train set (2/3) and a test set (1/3)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state
      = 42, test_size = 0.3)
```

A wide range of parameters was set in the dictionary ‘param_grid’. The code is as follows:

```
# Define the search parameter values
param_grid = {'max_depth': [2, 3, 4, 5, 6], 'min_samples_leaf': [1, 2,
      3, 4], 'min_samples_split': [1.0, 2, 3, 4], 'max_features': [1.0,
      0.8, 0.6, 0.5, 0.4, 0.3, 0.1,'auto',None]}
```

The parameter grid setting included parameters as keys and a list of parameter values as values. The parameter grid was searched to find the best values for the model. After setting the parameter values, the decision tree regression model was instantiated with the random_state number of 42, which meant every time it was run the output would remain the same. Next, the grid search method was instantiated with the required parameters and was fit with the train set. The code is as follows:

```
# Instantiate the model
est = DecisionTreeRegressor(random_state = 42)
# Instantiate and fit the grid search
gs_cv = GridSearchCV(est, param_grid, scoring = 'mean_squared_error',
      n_jobs = -1).fit(X_train, y_train)
```

Once the iterations were completed, the following code generated the optimal parameter values from the list of values:

```
# Best hyperparameter setting
print('Best Hyperparameters for Train set: %r' % gs_cv.best_params_)
```

The output is listed below.

```
Best Hyperparameters for train set: {'max_features': 0.8,
'min_samples_split': 2, 'max_depth': 4, 'min_samples_leaf': 2}
```

4.10.3.2. Developing and evaluating a decision tree regression predictive model

The 4 steps scikit-learn modeling was used on the decision tree regression model in the same way the previous models were built with the training set. The following block of codes was used to develop the decision tree regression model:

```
# Import classes
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
# Split the data set in a training set (2/3) and a test set (1/3)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state
      = 42, test_size = 0.3)
# Instantiate the decision tree regressor model
dtr_eval = DecisionTreeRegressor(random_state = 42, max_features = 0.8,
      min_samples_split = 2, max_depth = 4, min_samples_leaf = 2)
# Fit the model
dtr_eval.fit(X_train,y_train)
# Make the predictions on the train set
y_pred_train = dtr_eval.predict(X_train)
# Make the predictions on the test set
y_pred_test = dtr_eval.predict(X_test)
```

The following block of code calculated the performance measures by decision tree regression model on the training set:

```
# Find the error rate on the train set
rms_train = np.sqrt(mean_squared_error(y_train, y_pred_train))
mae_train = mean_absolute_error(y_train, y_pred_train)
r2_train = r2_score(y_train,y_pred_train)
print('Root Mean Squared Error:\t\t%0.2f'% rms_train)
print('Mean Absolute Error:\t\t\t%0.2f'% mae_train)
print('R2 Score:\t\t%0.2f'% r2_train)
```

The performance results of predictive model on the training set are listed in Table 29.

Table 29. The Performance Measures with Decision Tree Regression on Training Set

Performance Measures	Performance Scores
Root Mean Squared Error (RMSE)	3.47
Root Mean Squared Error (MAE)	2.17
R^2 Score	0.76

Again, the following block of code calculated the performance measures by the decision tree regression model on the test set.

```
# Find the error rate on test set
rms_test = np.sqrt(mean_squared_error(y_test, y_pred_test))
mae_test = mean_absolute_error(y_test, y_pred_test)
r2_test = r2_score(y_test, y_pred_test)
print('Root Mean Squared Error:\t\t%0.2f' % rms_test)
print('Mean Absolute Error:\t\t\t%0.2f' % mae_test)
print('R2 Score:\t\t%0.2f' % r2_test)
```

The performance results of predictive model on the test set are listed in Table 30. The high performance scores on the train and test sets indicated that the decision tree regression model was not a good fit for the problem on a revenue vehicle inventory data set and will not predict well on unseen data. Therefore, the decision tree regression model was not considered as our predictive model.

Table 30. The Performance Measures with Decision Tree Regression on Test Set

Performance Measures	Performance Scores
Root Mean Squared Error (RMSE)	3.48
Root Mean Squared Error (MAE)	2.20
R^2 Score	0.77

4.10.4. Comparison between random forest regression and gradient boosting regression model

Since the decision tree regression model was not be considered due to poor performance scores, the other two methods described above were compared for selection of the best predictive

model for service life on revenue vehicle inventory data. Table 31 shows the performance metric for both models.

Table 31. Comparisons of Performance Measures Between Random Forest Regression and Gradient Boosting Regression

Method	Full Training Data Set		
	RMSE	MAE	R^2 Score
Random Forest Regression	1.23	0.71	0.97
Gradient Boosting Regression	1.04	0.65	0.98

The above comparison results indicated that the gradient boosting regression model was a better fit for this problem. The RMSE score of 1.04 indicated that the prediction would fall within 1.04 below or above the standard deviation at 98% accuracy with a mean absolute error of 0.65 years of prediction difference from the actual service life of vehicles.

4.11. Building Gradient Boosting Regression Model for Service Life Prediction

Before applying the model on deployment data for predictions, the prediction of service life was compared with the actual service life of vehicles. The following code compared the vehicle's actual service life and the predicted service life:

```
# Get predicted service life
gbr_results = y.to_frame()
gbr_results['Prediction'] = y_gbr_pred
```

The output of the above code is listed below in Table 32. For simplicity, the output was shown with the first five values. The comparision between Service Life and Prediction shows that the model will perform well enough to deploy on unseen data. Futhermore, predictions could be even further improved by removing unnecessary features from the X variables.

Table 32. Comparison of Service Life vs. Predicted Service Life

Revenue Vehicle Inventory ID	Service Life	Prediction
24369.0	12.0	12.02986
24446.0	14.0	13.20664
48056.0	14.0	13.49888
42667.0	5.0	5.576827
48051.0	14.0	13.53025

The following code plotted the comparison histogram that showed the predicted service life vs. the actual service life:

```
# Create a list of service life and prediction
features_training = ['Service Life', 'Prediction']
ax = gbr_results.hist(column = features_training, figsize = (14, 6),
      bins = 50)
save_image('hist_gbr')
plt.show();
```

The above code produces the plots shown in Figure 21. The comparison histogram showed that the shape of the distribution of the data was normally distributed, and both were approximately bell-shaped. The range of the values was also same. Therefore, we could conclude that the model was performing well enough to predict the future service life of transit vehicles.

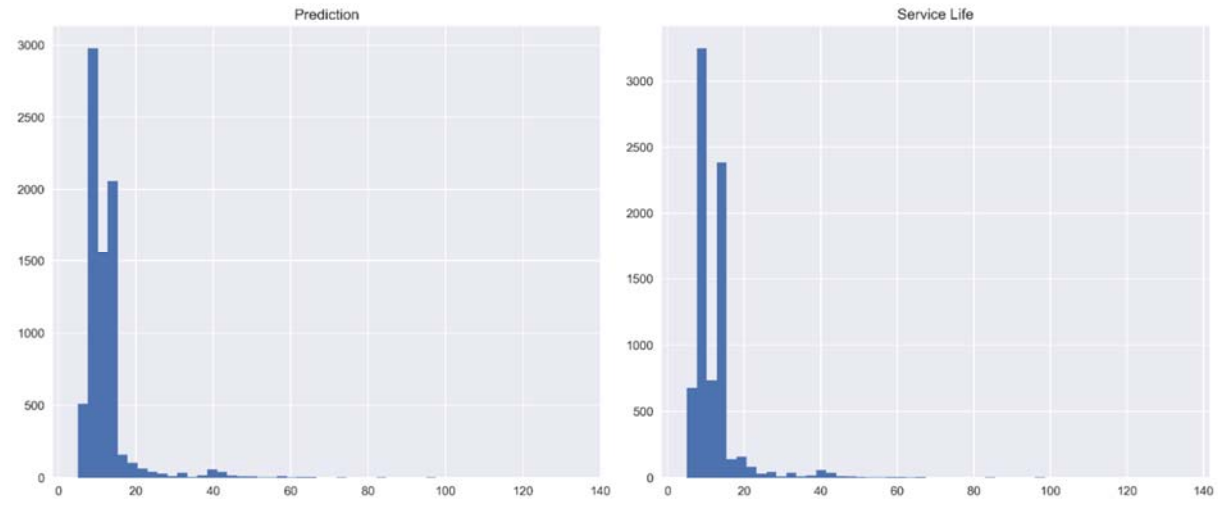


Figure 21. Comparison Histogram of Predicted Service Life vs. The Actual Service Life

The following code plots a regression line:

```
# Import classes
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(color_codes = True)
plt.subplots(figsize = (14, 7))
g = sns.regplot(x = gbr_results['Service Life'], y = y_gbr_pred, data =
      training)
regline = g.get_lines()[0]
regline.set_color('Cyan')
plt.title('Regression plot of Predicted Service Life')
plt.xlim(-5,150)
plt.ylim(-5,150)
```

The regression plot is shown in Figure 22. The regression line indicated the projected service life of transit vehicles.

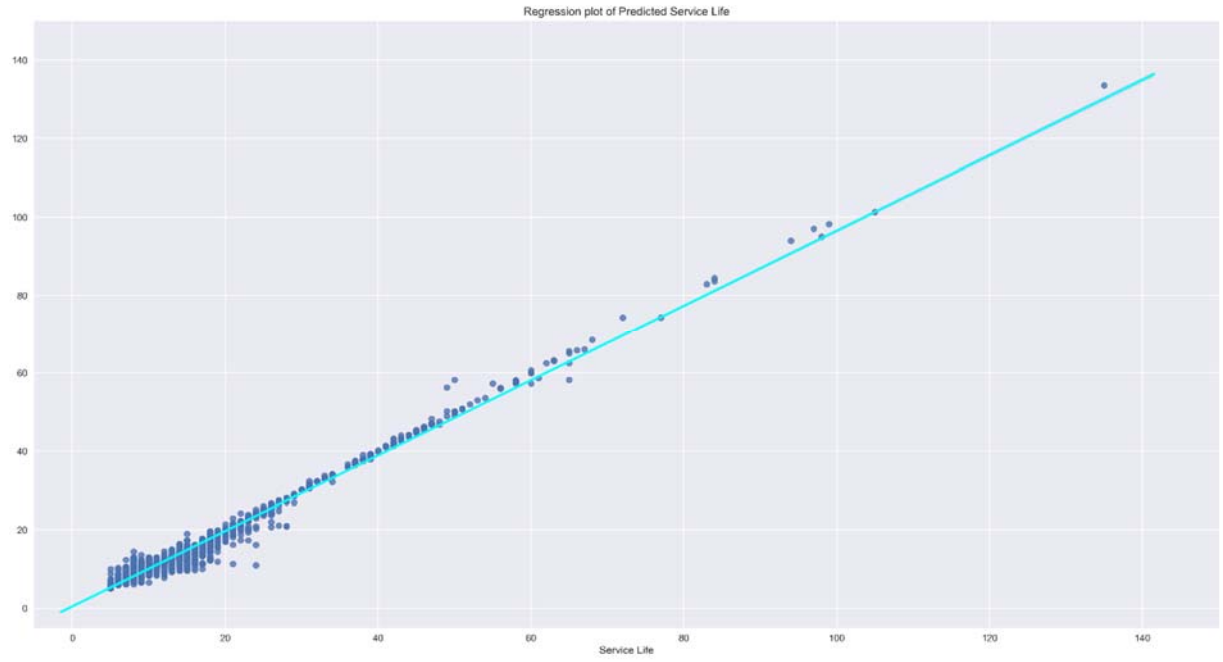


Figure 22. Regression Plot with a Regression Line of the Prediction of Service Life

4.11.1. Save the SGR predictive model

Since the model was trained and tested as well as the test data set already provided a good estimate of predictions errors, the model can perform even better if larger training data set can be used. The model generalizes and performs better if it is trained on the combined large data set (Downey, 2014). Therefore, the following predictive model was created on overall training data set and saved for unseen revenue vehicle inventory data for prediction. The following code saved the model in a pickle format so that the model can be used on deployment data for predictions.

```
# Import the class
from sklearn.externals import joblib
# Save the trained model to a file
joblib.dump(gbr, 'SGR_model_GBR.pkl')
```

4.12. Building a Gradient Boosting Regression Model with Feature Importance

In the previous gradient boosting regression predictive model, every useful features available in the data and some combined features were used in the training data set. It seemed reasonable to use as much information as available to build the model. However, sometimes some features may add redundant information which may lead poor generalization and some irrelevant features may cause overfitting the model. In addition, some poor features may return poor results. Sometimes, a large number of features may increase computation time without improving the regression model and may cause the problem on generalizing to train a model on a data set. As a result, a smaller set of most important features may produce better results. Therefore, in this model, 25% of the most important features were selected algorithmically. This process of selecting features is called feature selection, and this is very important to get better performance for any machine learning algorithms (Garreta & Moncecchi, 2013).

The gradient boosting regression can measure the feature importance by applying the feature_importances_ attribute after fitting the GradientBoostingRegressor. The following code ranks the top 30 most important features based on their respective importance measures:

```
# Display top 30 most important features
importances = pd.DataFrame({'Top 30 Important Features': X.columns,
      'importance': gbr.feature_importances_}).sort_values(by =
      'importance', ascending = False).reset_index(drop = True)
importances.head(30)
```

The output is listed in Table 33. The gradient boosting regression generates rank among the important features on a scale between 0 and 1 (Downey, 2014). The feature, VehicleLength_SeatingCapacity, is the top most important features with 8.7% importance score amongst all features.

Table 33. Top 30 Most Important Features and their Importance Scores

Index	Top 30 Most Important Features	Importance Scores
0	VehicleLength_SeatingCapacity	0.08783
1	ALMPAV_TFV	0.07824
2	Average Lifetime Miles per Active Vehicles	0.07704
3	Vehicle Length	0.06462
4	Vehicle Type_Service Life	0.05748
5	Total Miles on Active Vehicles During Period	0.05693
6	Seating Capacity	0.05662
7	ALMPAV_AFV	0.05176
8	StandingCap_SeatingCap	0.0492
9	Mode_Service Life	0.0468
10	TMOAVDP_AFV	0.04657
11	TMOAVDP_TFV	0.04514
12	VehicleLength_StandingCapacity	0.04153
13	Total Fleet Vehicles	0.03268
14	Standing Capacity	0.02562
15	Fuel Type_Service Life	0.01883
16	ADA Fleet Vehicles	0.01591
17	Active Fleet Vehicles	0.01472
18	Fuel Type_Electric Propulsion Power	0.00988
19	Funding Source_UA	0.00903
20	RebuildYear_ManufactureYear	0.00814
21	Fuel Type_Diesel Fuel	0.00746
22	Emergency Contingency Vehicles	0.00696
23	Funding Source_NFPA	0.00679
24	Funding Source_OF	0.00658
25	TOS_PT	0.00643
26	TOS_DO	0.00617
27	Vehicle Type_Bus	0.00457
28	Ownership Type_OOPA	0.00405
29	Rebuild Year	0.00386

After ranking the top 30 most important features, a plot was created based on their relative importance with the top 30 most important features. The following code shows these top 30 most important features in a bar chart:

```
# Display important features in bar graph
importances.head(30).plot(kind = 'bar', figsize = (14, 8) ,use_index =
      'name', x = 'Top 30 Important Features')
plt.title('Top 30 Important Features and importance score')
```

```
plt.ylabel('Importance Score')
save_image('important_features')
plt.show()
```

The plot with top 30 most important features is shown in Figure 23 highlighted the top most 30 features which was ranked by the relative feature importance for gradient boosting regression predictive model. The relative importance of features indicates how much a feature can contribute predicting a target variable. The greater feature's importance means the feature is being used more often. Since gradient boosting regression is an ensemble tree model, the scores are averaged for each feature across all trees, and the sum of all important features is equal to 1. In this gradient boosting regression predictive model, the relative feature importance for top ten features were most significant and accounted for about 60% of total feature importance. Similarly, the top five most important features contributed about 35% of relative feature importance. There were only two internal features ranked amongst the top five important features, and the most important feature is VehicleLength_SeatingCapacity used in this model (Johnson, et al., 2017). Therefore, we can conclude that the creation of new features by combining the different combination of features have the significant impact on the model.

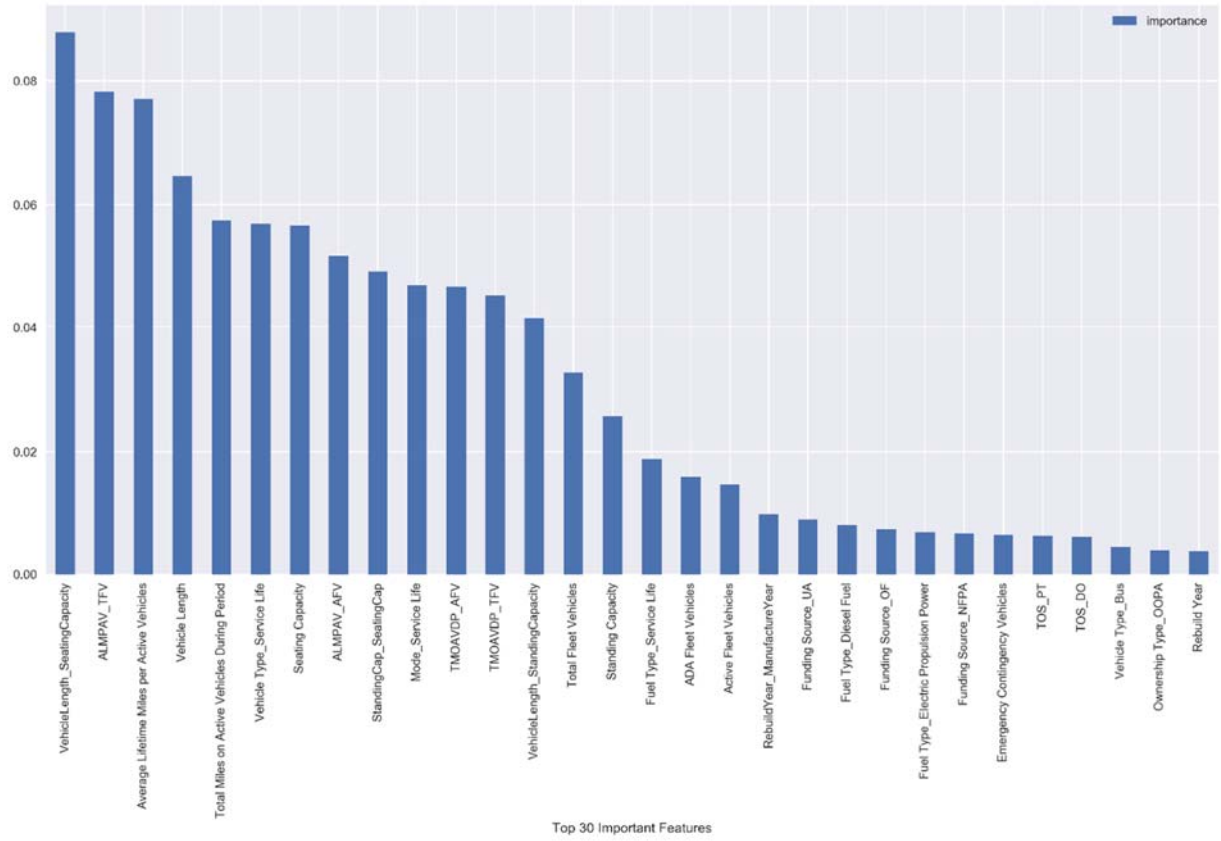


Figure 23. Bar Plot with Top 30 Important Features and Importance Score

Next, the top 30 features were listed in the variable object which were used to build the model. The below code listed the top 30 most important features.

```
# List of top 30 most important features
important_features = ['Vehicle Length', 'Average Lifetime Miles per
      Active Vehicles', 'Total Miles on Active Vehicles During Period',
      'Seating Capacity', 'Standing Capacity', 'ADA Fleet Vehicles',
      'Active Fleet Vehicles', 'Emergency Contingency Vehicles', 'Total
      Fleet Vehicles', 'Fuel Type_Service Life', 'Mode_Service Life',
      'Vehicle Type_Service Life', 'VehicleLength_SeatingCapacity',
      'RebuildYear_ManufactureYear', 'VehicleLength_StandingCapacity',
      'StandingCap_SeatingCap', 'TMOAVDP_TFV', 'TMOAVDP_AFV',
      'ALMPAV_TFV', 'ALMPAV_AFV', 'TOS_DO', 'TOS_PT', 'Mode_CC',
      'Supports_Mode_MB', 'Fuel Type_Diesel Fuel', 'Vehicle Type_Bus',
      'Ownership Type_OOPA', 'Funding Source_NFPA', 'Funding
      Source_OF', 'Funding Source_UA']
```

The following block of codes was used to build the predictive model using 30 most important features. At first, the top 30 most important features were stored in X variables on the revenue vehicle inventory data. Then, the Scikit-learn modeling patterns were applied to build the model.

```
# Store feature matrix in 'X'
X_imp = X[important_features]
# Store response vector in 'y'
y_imp = y
# Import class
from sklearn.ensemble import GradientBoostingRegressor
# Instantiate regression model with tuned hyperparameters using least-
# squares
gbr_imp = GradientBoostingRegressor(n_estimators = 3000, max_features =
      0.6, min_samples_split = 4, learning_rate = 0.01, max_depth = 10,
      min_samples_leaf = 3, loss = 'ls')
# Fit regression model to the overall training data set
gbr_imp.fit(X_imp, y_imp)
# Make prediction on Overall training data set
y_pred_imp = gbr_imp.predict(X_imp)
```

In order to check the error rates and other performance measures on the split training set, the following block of code was used:

```
# Find the error rate on the full data set
rms_imp = np.sqrt(mean_squared_error(y_imp, y_pred_imp))
mae_imp = mean_absolute_error(y_imp, y_pred_imp)
r2_imp = r2_score(y_imp,y_pred_imp)
print('Root Mean Squared Error:\t\t%0.2f'% rms_imp)
print('Mean Absolute Error:\t\t\t%0.2f'% mae_imp)
print('R2 Score:\t\t%0.2f'% r2_imp)
```

Finally, the performance results of predictive model on the train set are listed in Table 34. In the result, the root mean squared error of 0.83 and the R^2 score of 0.99 indicates that the predictions were fallen less than 1 year below or above the standard deviation with 99% accuracy rate and a mean absolute error of 0.45 for predictions.

Table 34. The Performance Measures by Gradient Boosting Regression with Top 30 Most Important Features on Full Data Set

Performance Measures	Performance Scores
Root Mean Squared Error (RMSE)	0.83
Root Mean Squared Error (MAE)	0.45
R^2 Score	0.99

The performance results were compared between gradient boosting regression model with and without top 30 most important features shown in Table 35. The comparison results showed that the gradient boosting model with top 30 important features produced the better model.

Table 35. Comparison of Performance Results Between Gradient Boosting Regression Model and Gradient Boosting Regression Model with Top 30 Important Features

Method	Full Training Data Set		
	RMSE	MAE	R^2 Score
Gradient Boosting Regression	1.04	0.65	0.98
Gradient Boosting Regression with Top 30 Most Important Features	0.83	0.45	0.99

The following block of code was used to compare the predicted service life of vehicles with the actual service life of the same vehicles after applying only the 30 most important features of gradient boosting regression model (only 5 rows shown):

```
# Display predictions
results_imp = y_imp.to_frame()
results_imp['Prediction'] = y_pred_imp
# Show first 5 rows
results_imp.head()
```

The output of the above code is shown in Table 36. The comparison showed a very close predicted service life with the actual service life of vehicles.

Table 36. Predicted Service Life vs. Actual Service Life by Top 30 Most Important Features

Revenue Vehicle Inventory ID	Service Life	Prediction
24369.0	12.0	12.13049
24446.0	14.0	13.84231
48056.0	14.0	13.73627
42667.0	5.0	5.201973
48051.0	14.0	13.84958

The following block of code plots a comparison histogram that shows predicted service life versus the actual service life with the gradient boosting regression model using top 30 most important features:

```
# List features
features = ['Service Life', 'Prediction']
results_imp.hist(column = features, figsize = (14, 6), bins = 50)
save_image('hist_imp')
plt.show();
```

The above code generates histograms shown in Figure 24. The comparison histogram showed that the shape of the distribution of the data in both plots is normally distributed. The range of the values was also same. Therefore, we could say that the model was performing well enough to predict the future service life of vehicles using gradient boosting regression model with top 30 most important features.

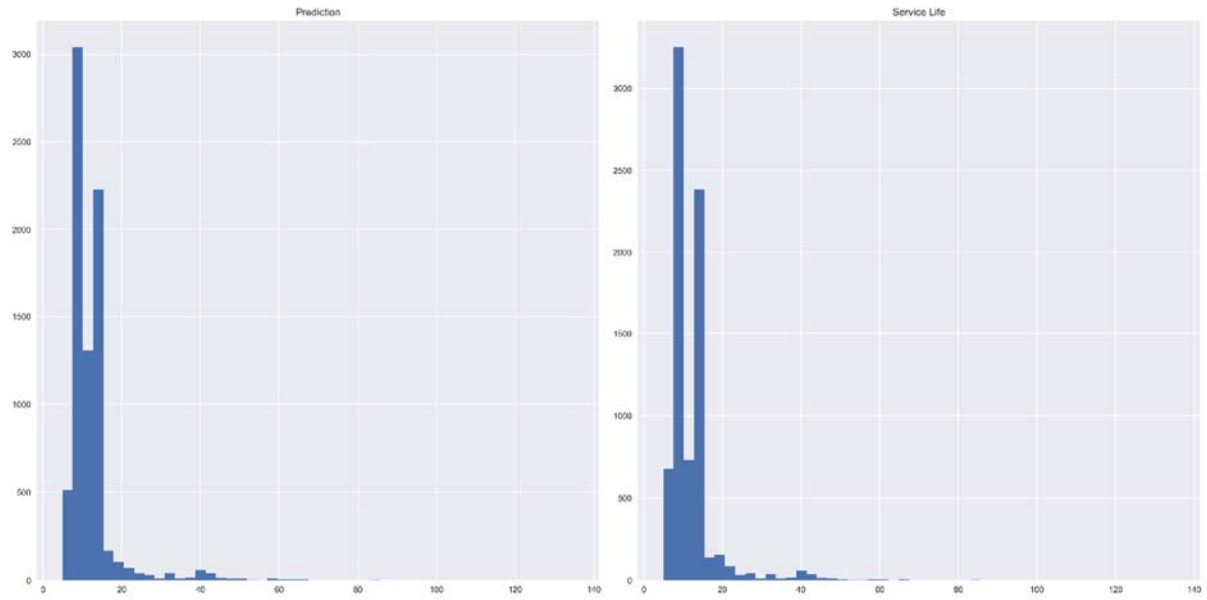


Figure 24. Comparison Histogram of Prediction vs. Actual Service Life

The following code was used on revenue vehicle inventory data in order to check the regression line with top 30 most important features:

```
# Import classes
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(color_codes = True)
plt.subplots(figsize = (14,7))
g = sns.regplot(x = results_imp['Service Life'], y = y_gbr_pred, data =
      training)
regline = g.get_lines()[0]
regline.set_color('Cyan')
plt.title('Regression plot of Predicted Service Life with top 30
      features')
plt.xlim(-5,150)
plt.ylim(-5,150)
save_image('reg_imp')
plt.show()
```

The above code plotted a regression plot with a line which was a prediction for the service life of vehicles shown in Figure 25.

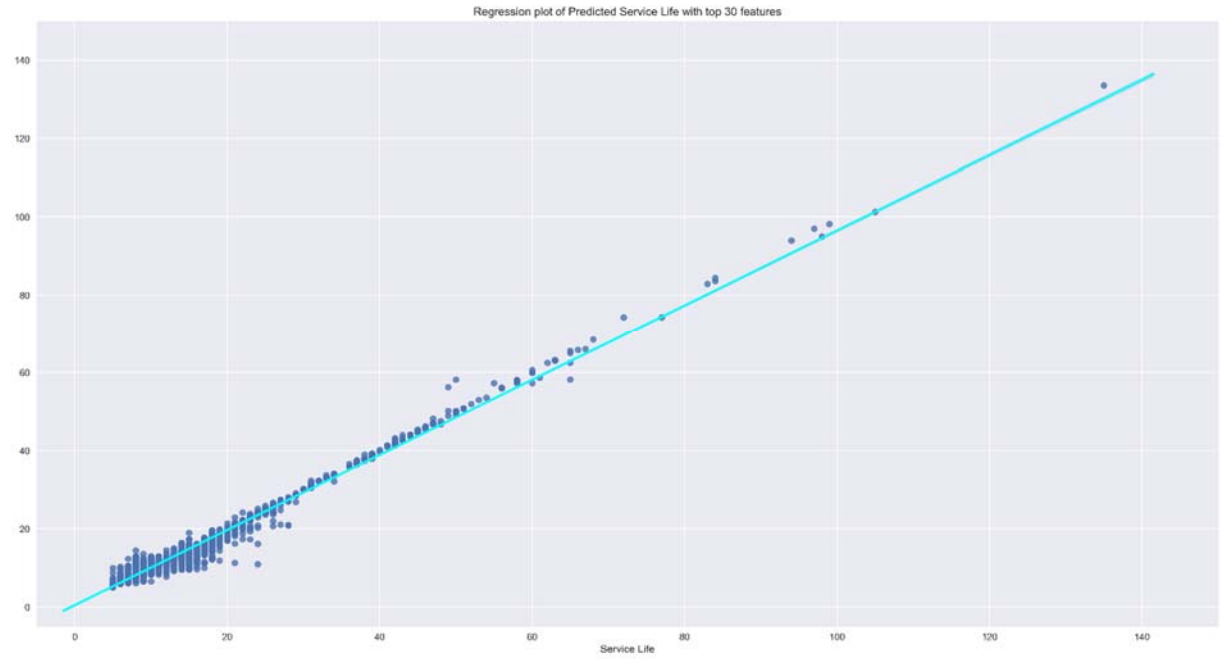


Figure 25. Regression Plot of Predicted Service Life vs. Actual Service Life with Top 30 Most Important Features

4.13. Comparison Analysis of Predictions

Since we got the comparison results of predicted service life and actual service life for both on gradient boosting regression model with all the features and with 30 most important features, the results are inserted in Table 37 for comparisons. The prediction results in the below table indicated the predictions were almost close in both cases. However, removing redundant features from the model improved the performance of the model. Therefore, the predictive model by gradient boosting regression model with top 30 most important features were chosen to solve the state of good repair problem.

Table 37. Comparison of Actual Service Life vs. Predicted Service Life with All Features and Top 30 Important Features

RVI ID	Service Life	Prediction (With All Features)	Prediction (With top 30 important features)
24369	12	12.0299	12.1305
24446	14	13.2066	13.8423
48056	14	13.4989	13.7363
42667	5	5.57683	5.20197
48051	14	13.5303	13.8496

4.14. Save the Gradient Boosting Regression Model with Top 30 Important Features

At this point, the model was saved in a pickle format for predictions to deployment data.

```
# Save the trained model with top 30 important features to a file
joblib.dump(gbr_imp, 'SGR_model_imp.pkl')
```

4.15. Make Predictions on Deployment Data

Since the gradient boosting regression model was developed with top 30 most important features, the model was loaded to apply to the deployment data set for predictions. The following codes loaded the desired model and the necessary data set for predictions:

```
# Load the model that was trained previously
SGR_model = joblib.load('SGR_model_imp.pkl')
# Load cleaned deployment data for machine learning predictive model
X_deploy = pd.read_csv('..//NTD/Final Deployment
      Data.csv').set_index('Revenue Vehicle Inventory ID')
# Load non-retired revenue vehicle inventory data since 2008 for
# prediction
revenue_all_vehicles =
      pd.read_csv('..//NTD/Revenue_Vehicle_Inventory_all_years.csv').se
      t_index('Revenue Vehicle Inventory ID').drop(['Unnamed: 0'], axis
      = 1)
```

The shape attribute was applied on all data to see the total number of vehicles, and it was further applied on the deployment data to see the number of non-retired vehicles. The block of code is as follows:

```
# Show number of rows and columns
revenue_all_vehicles.shape
# Show number of rows and columns
X_deploy.shape
```

The above code showed 42440 vehicles were in the revenue vehicle inventory database from 2008 to 2016. Among them, 31146 vehicles were still in operation which needed to be predicted when their service life would be expired. The following code found the number of vehicles which were missing with Manufacture Year.

```
# Find the number of vehicles which are missing with 'Manufacture Year'
# data
revenue_all_vehicles['Manufacture Year'].isnull().sum()
```

The above code showed that 3189 vehicles were missing with ‘Manufacture Year’ information, and thus these data were not included in the deployment data set.

The following code finds the number of vehicles with missing Fuel Type.

```
# Find the number of vehicles which are missing with 'Fuel Type'
# information
revenue_all_vehicles['Fuel Type'].isnull().sum()
```

The above code showed that 14824 vehicles did not have fuel type information. This information needs to be brought to attention to transit agencies so that they can update revenue vehicle inventory data with fuel type information.

Now, the following code made the predictions on the deployment data set and created a new column Predicted Service Life in the data frame.

```
# Make the predictions on the non-retired data
y_pred = SGR_model.predict(X_deploy)
# Create a column 'Predicted Service Life' with the prediction
X_deploy['Predicted Service Life'] = y_pred
```

Since the model generated the predicted service life value for deployment data, the newly created column was merged with all revenue inventory data by performing a join operation. The merged data frame was stored in a new data frame. The code is as follows:

```
# Merge two DataFrame by a join operation
revenue_all = revenue_all_vehicles.join(X_deploy['Predicted Service
      Life'], how = 'right')
```

Now, the following code created a new column of Projected Retired Year by adding Predicted Service Life with the Manufacture Year. The code is as follows:

```
# Create a new column by adding Predicted Service Life with Manufacture
# Year
revenue_all['Projected Retired Year'] = (revenue_all['Manufacture
      Year'] + (revenue_all['Rebuild Year'] - revenue_all['Manufacture
      Year']).fillna(0) + revenue_all['Predicted Service Life'] +
      1.0).round()
```

Finally, the model was saved in a comma separated CSV file as a final report for use as a guide for predictions by transit agencies and the FTA. The code is as follows:

```
# Save the result
revenue_all.to_csv('Report_Non-Retired Revenue Vehicle Inventory 2008-
      2016 Results.csv', sep = ',')
```

4.16. The Deployment Data Analysis

Before doing any data analysis with the deployment data, some column names were renamed by adding the underscore (_) between words for easy manipulation. The code is as follows:

```
# Rename columns
df = revenue_all.rename(columns = {'Vehicle Type': 'Vehicle_Type',
      'Predicted Service Life': 'Predicted_Service_Life', 'Projected
      Retired Year': 'Projected_Retired_Year', 'Vehicle Length':
      'Vehicle_Length', 'Fuel Type': 'Fuel_Type'})
```

The following code checked the number of vehicles in the deployment data based on each vehicle type by running value_counts() function:

```
# Count the number of vehicles by vehicle type
df.Vehicle_Type.value_counts()
```

The output is shown in Table 38. The resulting output is in descending order where the Cutaway is the most frequently occurring and the Inclined Plane Vehicle is the least frequently occurring vehicles.

Table 38. Number of Vehicles in Deployment Data by Vehicle Type

Vehicle Type	Number of Vehicles
Cutaway	11470
Bus	8729
Van	5419
Minivan	2616
Over-the-road Bus	692
Automobile	526
Articulated Bus	294
Commuter Rail Passenger Coach	234
Heavy Rail Passenger Car	210
Sports Utility Vehicle	196
Ferryboat	172
Light Rail Vehicle	133
Commuter Rail Locomotive	121
Commuter Rail Self-Propelled Passenger Car	91
School Bus	72
Other	56
Vintage Trolley	39
Cable Car	16
Streetcar Rail	16
Trolleybus	12
Double Decker Bus	12
Automated Guideway Vehicle	9
Aerial Tramway	8
Inclined Plane Vehicle	3

The following code can further visualize the above number in a horizontal bar plot:

```
# Plot the number of vehicles by vehicle type
```

```
df.Vehicle_Type.value_counts(ascending = True).plot(kind = 'barh',
      figsize = (12, 5))
plt.title('Number of Vehicles by Vehicle Type')
plt.xlabel('Number of Vehicles')
plt.ylabel('Vehicle Type')
save_image('bar_vt')
```

The bar plot of vehicle count by type is shown in Figure 26.

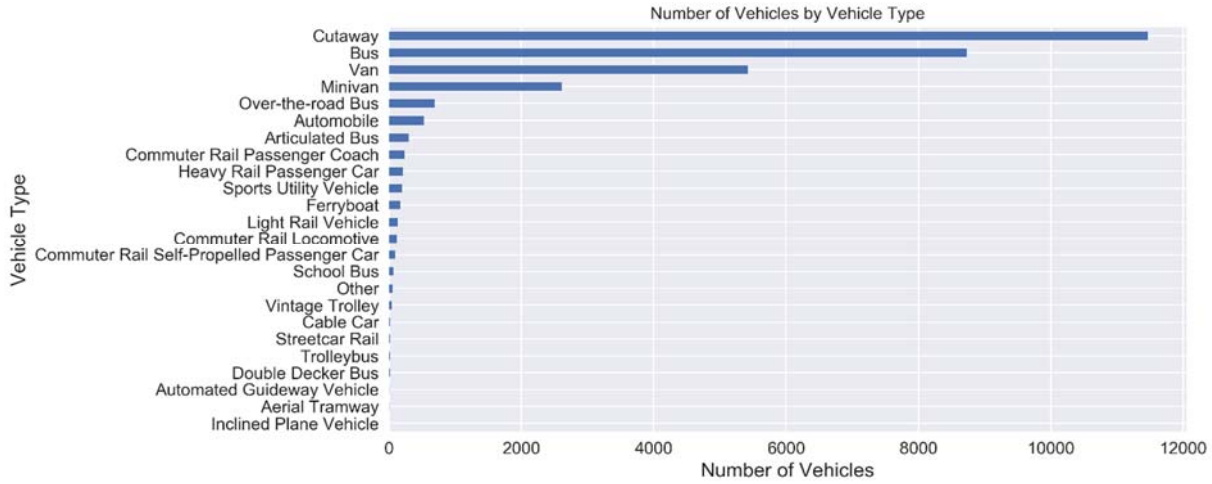


Figure 26. Bar Plot of Vehicle Count by Vehicle Type

Next, statistical analysis was performed on the predicted service life by vehicle type using *agg()* function to visualize the average service life by vehicle type. The following code plotted the statistical analysis:

```
# Plot statistical analysis of average predicted service life by
# vehicle type
df.groupby('Vehicle_Type').Predicted_Service_Life.agg(['min', 'mean',
      'max']).plot(kind = 'barh', figsize = (14, 6))
plt.title('Average Predicted Service by Vehicle Type')
plt.xlabel('Average Predicted Service Life')
plt.ylabel('Vehicle Type');
save_image('mean_bar')
```

The bar plot is as shown in Figure 27.

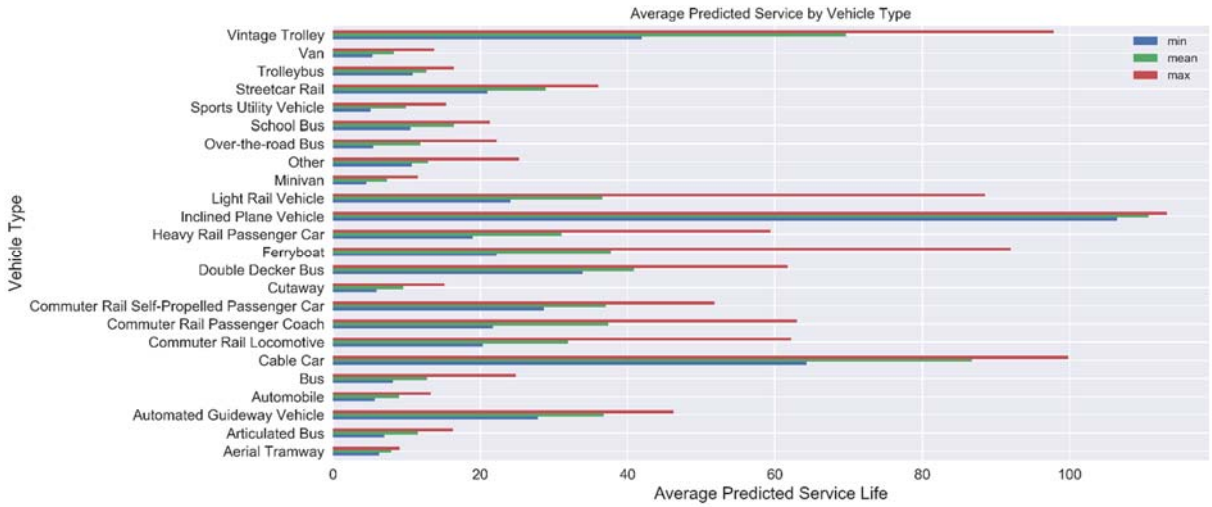


Figure 27. Bar Plot of Statistical Analysis of Predicted Service Life by Vehicle Type

4.16.1. Cross Tabulation Analysis

A cross-tabulation analysis, also known as contingency table analysis, is a table shows the frequency distribution of one variable in rows and another one in columns (Contingency table, 2018). A typical cross-tabulation table comparing the two variables Fuel Type with Vehicle Mode is shown below:

```
# Create cross tabulation on vehicle type by mode
pd.crosstab(df.Vehicle_Type, df.Mode, margins = True)
```

The output is shown in Table 39. The table showed the distribution of a Vehicle Type with Mode. A few vehicle type had a single mode, however; most of the vehicle type had multiple mode.

Table 39. Contingency Table of Vehicle Type by Vehicle Model

Vehicle Type	Mode																		All
	AR	CB	CC	CR	DR	DT	FB	HR	IP	LR	MB	MG	RB	SR	TB	TR	VP	YR	
Aerial Tramway	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	8
Articulated Bus	0	30	0	0	0	0	0	0	0	0	239	0	25	0	0	0	0	0	294
Automated Guideway Vehicle	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	9
Automobile	0	0	0	0	499	10	0	0	0	0	17	0	0	0	0	0	0	0	526
Bus	0	452	0	0	1700	0	0	0	0	0	6555	0	18	0	0	0	4	0	8729
Cable Car	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16
Commuter Rail Locomotive	9	0	0	112	0	0	0	0	0	0	0	0	0	0	0	0	0	0	121
Commuter Rail Passenger Coach	20	0	0	213	0	0	0	0	0	0	0	0	0	0	0	0	0	1	234
Commuter Rail Self-Propelled Passenger Car	0	0	0	86	0	0	0	0	0	0	0	0	0	0	0	0	0	5	91
Cutaway	0	302	0	0	8266	0	0	0	0	0	2902	0	0	0	0	0	0	0	11470
Double Decker Bus	0	2	0	0	0	0	0	0	0	0	9	0	1	0	0	0	0	0	12
Ferryboat	0	0	0	0	0	0	172	0	0	0	0	0	0	0	0	0	0	0	172
Heavy Rail Passenger Car	0	0	0	0	0	0	0	210	0	0	0	0	0	0	0	0	0	0	210
Inclined Plane Vehicle	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	3
Light Rail Vehicle	0	0	0	0	0	0	0	0	0	98	0	0	0	33	0	0	0	2	133
Minivan	0	0	0	0	2433	4	0	0	0	0	31	0	0	0	0	0	148	0	2616
Other	0	23	0	0	5	0	1	0	0	0	9	0	7	0	0	0	11	0	56
Over-the-road Bus	0	513	0	0	0	0	0	0	0	0	179	0	0	0	0	0	0	0	692

Table 39. Contingency Table of Vehicle Type by Vehicle Model (continued)

Vehicle Type	Mode																		All
	AR	CB	CC	CR	DR	DT	FB	HR	IP	LR	MB	MG	RB	SR	TB	TR	VP	YR	
School Bus	0	0	0	0	62	0	0	0	0	0	10	0	0	0	0	0	0	0	72
Sports Utility Vehicle	0	0	0	0	135	1	0	0	0	0	0	0	0	0	0	0	60	0	196
Streetcar Rail	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	16
Trolleybus	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	12
Van	0	0	0	0	3158	7	0	0	0	0	313	0	0	0	0	0	1941	0	5419
Vintage Trolley	0	0	0	0	0	0	0	0	0	5	0	0	0	34	0	0	0	0	39
All	29	1322	16	411	16258	22	173	210	3	103	10264	9	51	83	12	8	2164	8	31146

4.17. Analysis on the Condition of Buses Based on Predicted Service Life

The following code was used to analyze the predicted retirement years for buses. The code divided nationwide bus data into two sets. One set included bus data which predicted retirement years until 2017, and other set included bus data which predicted retirement after the year 2017.

```
# Set plot size
fig, ax = plt.subplots(1, 2, figsize = (14, 5))
# Set first plot
df_bus_ = df.loc[(df.Vehicle_Type == 'Bus') &
      (df.Projected_Retired_Year < = 2017)]
df_bus_.Projected_Retired_Year.plot.hist(ax = ax[0], color = 'red')
# Give the plot a main title
ax[0].set_title('Buses already Retired by prediction by previous
      years')
# Set text for the x axis
ax[0].set_xlabel('Predicted Retired Years')
# Set text for y axis
ax[0].set_ylabel('Number of Buses')
# Second plot
df_bus = df.loc[(df.Vehicle_Type == 'Bus') & (df.Projected_Retired_Year
      > = 2018)]
df_bus.Projected_Retired_Year.plot.hist(ax = ax[1])
# Give the plot a main title
ax[1].set_title('Buses will be Retired in future Years')
# Set text for the x axis
ax[1].set_xlabel('Predicted Retired Years')
# Set text for y axis
ax[1].set_ylabel('Number of Buses');
save_image('bus_analysis')
```

The above code produces the current conditions of buses shown in Figure 28. The plot showed that 1983 buses out of 8729 buses which were about 23% of buses nationally already predicted to be retired and needed immediate attention to either replace or rehabilitate.

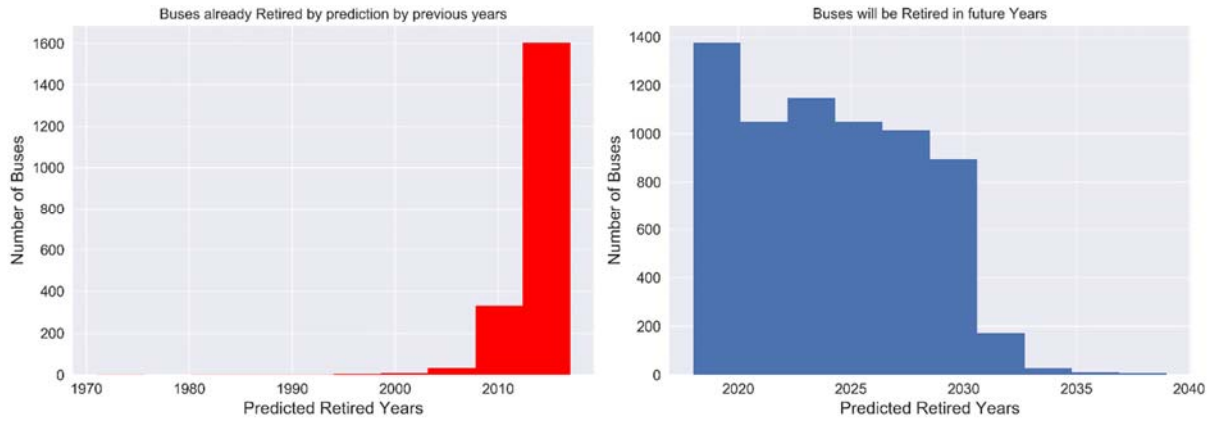


Figure 28. The Condition of Buses Based on Predicted Service Life

4.18. Data Analysis on Fargo Metropolitan Area Transit (MAT Bus) data

Now, we will analyze the condition of transit vehicles in a small urban transit agency as an example of how the transit agency can get the benefit of using the model. For this purpose, the transit agency, Fargo Metropolitan Area Transit (MAT Bus), was chosen which had 19 vehicles in their fleet. The following code stores the Fargo Metropolitan Area Transit (MAT Bus) agency data to fargo_mat data frame.

```
# Filter data by agency name
fargo_mat = df[(df['Agency Name'] == 'City of Fargo, DBA: Metropolitan
      Area Transit')]
```

The following code shows the number of transit vehicles in MAT Bus.

```
# Count the vehicles
fargo_mat.shape
```

The shape attribute shows that 19 vehicles are in operation in Fargo, North Dakota area by Metropolitan Area Transit. The following code shows the number of each type of vehicles operated by the agency.

```
# Count the number of vehicles by vehicle type
fargo_mat.Vehicle_Type.value_counts()
```

The output is shown in Table 40 showed the MAT Bus currently held 11 buses and 8 cutaways in their fleet.

Table 40. Number of Vehicles by Vehicle Type at MAT Bus

Type of Vehicles	Number of Vehicles
Bus	11
Cutaway	8

The following block of code shows the number of vehicles by vehicle type in a bar plot.

```
# Plot the number of vehicles by vehicle type
f, ax = plt.subplots(figsize = (12, 4))
sns.countplot(y = 'Vehicle_Type', data = fargo_mat)
ax.set_title('Number of Vehicles by Vehicle Type')
ax.set_xlabel('Number of Vehicles')
ax.set_ylabel('Vehicle Type');
save_image('mat_counts')
```

The above code will plot a bar graph shown in Figure 29.

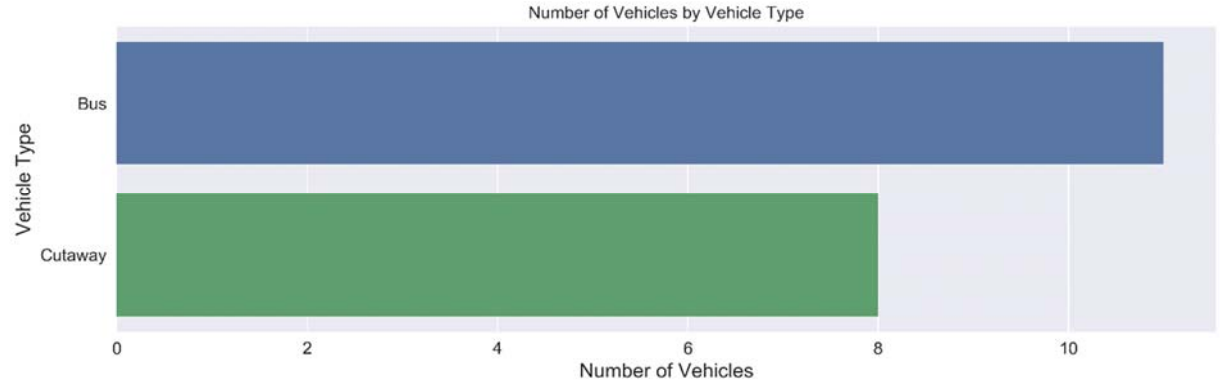


Figure 29. Bar Plot of the Number of Vehicles by Vehicle Type at MAT Bus

The statistical analysis was performed as well on the predicted service life by the following code:

```
# Statistical analysis of service life by vehicle type
fargo_mat.groupby('Vehicle_Type').Predicted_Service_Life.agg(['count',
      'min', 'max', 'mean'])
```

The output is shown in Table 41. The statistical analysis showed the minimum, the maximum, and the average service life of the vehicle by vehicle type. The predicted average service life is very much close to the default useful life specified by the FTA which is 14 for bus and 10 for cutaways.

Table 41. Statistical Analysis of Service Life by Vehicle Type on MAT Bus

Vehicle Type	count	min	max	mean
Bus	11	10.31104	16.362	12.85894
Cutaway	8	8.561481	10.85107	9.418001

The following code further visualizes the statistical analysis.

```
# Plot statistical analysis of predicted service life by vehicle type
fargo_mat.groupby('Vehicle_Type').Predicted_Service_Life.agg(['mean',
      'min', 'max']).plot(kind = 'barh', figsize = (14, 4))
plt.title('Statistical Analysis of Predicted Service Life of Vehicles
      by Vehicle Type')
plt.xlabel('Number of Vehicles')
plt.ylabel('Vehicle Type');
save_image('mat_sa')
```

The above code plotted the statistical analysis shown in Figure 30.

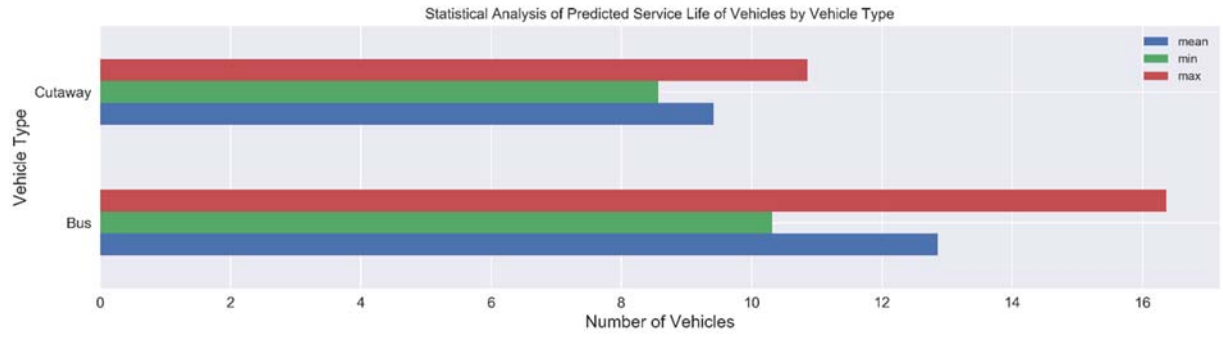


Figure 30. Bar Plot of Statistical Analysis of Predicted Service Life by Vehicle Type on MAT Bus

In order to see the overall condition of transit vehicles operated by Metropolitan Area Transit, the revenue vehicle data was filtered with predicted retired year until 2018 and predicted retired year after 2018. The following block of codes filters out data and plots into two subplots.

```
# Set plot size
fig, ax = plt.subplots(1, 2, figsize = (14, 5))
df_fargo_mat_ = fargo_mat.loc[fargo_mat.Projected_Retired_Year < =
      2017]
df_fargo_mat_.Projected_Retired_Year.plot.hist(ax = ax[0], color =
      'red')
ax[0].set_title('Fargo MAT Vehicles already Retired by prediction by
      previous years')
ax[0].set_xlabel('Predicted Retired Years')
ax[0].set_ylabel('Number of Vehicles')
df_fargo_mat = fargo_mat.loc[fargo_mat.Projected_Retired_Year > = 2018]
df_fargo_mat.Projected_Retired_Year.plot.hist(ax = ax[1])
ax[1].set_title('Fargo MAT Vehicles will be Retired in future Years')
ax[1].set_xlabel('Predicted Retired Years')
ax[1].set_ylabel('Number of Vehicles');
save_image('fargo_mat_analysis')
```

The above block of code plots the condition of vehicles for MAT Bus shown in Figure 31.

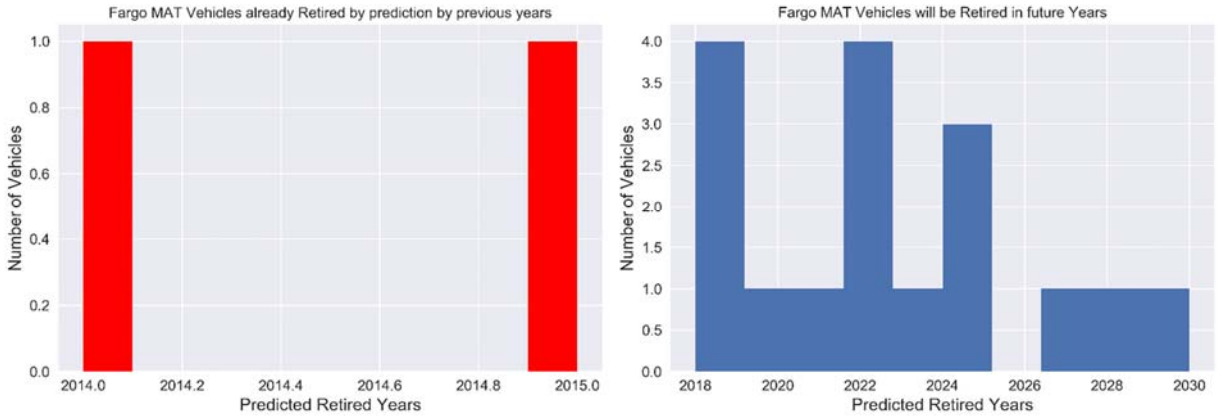


Figure 31. MAT Bus Projected Retired Year

The following code calculates the number of vehicles which are predicted to be retired before the year 2018.

```
# Show number of rows and columns
df_fargo_mat_.shape
```

The above code shows that 2 out of 19 vehicles which are about 11% vehicles need to be replaced or rehabilitated immediately. The following code shows the predicted retired year of each vehicle for MAT Bus.

```
# Projected retired year for MAT Bus by vehicle type
df_fargo_mat[['Projected_Retired_Year', 'Vehicle_Type']]
```

The code prints the predicted retired year shown in Table 42.

Table 42. The Projected Retired Year for MAT Bus

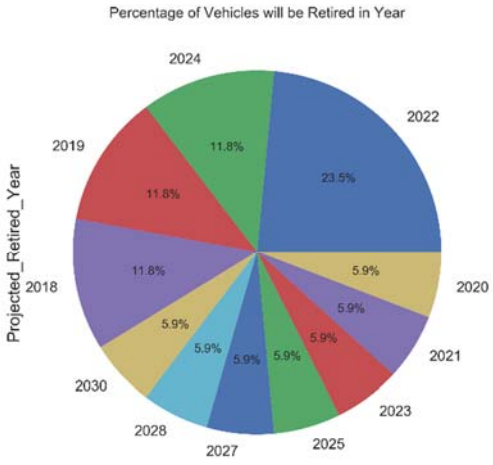
Revenue Vehicle Inventory ID	Projected Retired Year	Vehicle Type
13492	2018	Bus
24444	2019	Bus
30530	2021	Bus
38184	2019	Cutaway
38186	2020	Cutaway
38188	2022	Bus
43198	2024	Bus
47932	2023	Bus
47933	2022	Bus
53628	2022	Cutaway
59603	2022	Cutaway
59604	2028	Bus
337297	2024	Cutaway
337314	2018	Cutaway
343269	2025	Cutaway
343303	2030	Bus

The following block of code plots a pie chart and a table with projected retired year.

```
# Plot chart
plt.figure(figsize = (14,6))
ax1 = plt.subplot(121, aspect = 'equal')
fargo_mat_ret = df_fargo_mat.Projected_Retired_Year.astype
      (int).value_counts().plot (kind = 'pie', autopct = '%1.1f%%')
fargo_mat_ret.set_title('Percentage of Vehicles will be Retired in
      Year')
# Plot table
fargo_mat_tbl =
      df_fargo_mat.Projected_Retired_Year.astype(int).value_counts()
```

```
# Import class
from pandas.tools.plotting import table
ax2 = plt.subplot(122)
plt.axis('off')
tbl = table(ax2, fargo_mat_tbl, loc = 'center', colWidths = [0.3])
tbl.auto_set_font_size(False)
tbl.set_fontsize(14)
save_image('fargo_mat_pie')
plt.show();
```

The plot with table is shown in Figure 32. The pie chart showed the percentage of vehicles and the table showed the corresponding number of vehicles which will be retired in the future year. Therefore, the Metropolitan Area Transit (MAT Bus) should be aware of the condition of their transit vehicles and plan for replacement.



Projected_Retired_Year	
2022	4
2024	2
2019	2
2018	2
2030	1
2028	1
2027	1
2025	1
2023	1
2021	1
2020	1

Figure 32. Pie Chart and Table to Show the Projected Retired Year on MAT Bus

4.19. Make Prediction on Any Single Vehicle

The gradient boosting regression model with top 30 most important features was also used in order to predict any single vehicle. After the initial setup with parameters, the model and other necessary CSV files were loaded. The code for loading model and data is as follows:

```
# Load the predictive model
model = joblib.load('SGR_model_imp.pkl')
```

```
# Load the model that was trained previously with only top 30 important
# features
revenue_all = pd.read_csv
      ('..//NTD/Revenue_Vehicle_Inventory_all_years.csv').drop(['Unname
      d: 0'], axis = 1)
# Load the cleaned deployment data
X_deploy = pd.read_csv('..//NTD/Final Deployment Data.csv')
```

Since the data analysis on Metropolitan Area Transit (MAT Bus), Fargo, North Dakota was performed by gradient boosting regression predictive model, a single vehicle was chosen from the MAT Bus for further analysis by gradient boosting regression predictive model with the top 30 important features. For this analysis, a bus of vehicle inventory id of 24444 was chosen to predict its projected retired year and compare the result with the previous analysis. The following code was used to select the vehicle store it to a data frame:

```
# Select vehicle with RVI ID of 24444
vehicle_24444 = X_deploy[X_deploy['Revenue Vehicle Inventory ID'] ==
      24444]
```

The selected vehicle is listed in Table 43. Since the processed data had 119 columns, a few columns were entered here for simplicity.

Table 43. Processed Columns on Revenue Vehicle Data for Machine Learning Algorithm

Revenue Vehicle Inventory ID	Seating Capacity	Standing Capacity	Vehicle Length	…	Vehicle Type_Bus	TOS_PT	Mode_Service Life
24444.0	16	0	25	…	True	1	11.0

The following code was used to show the necessary vehicle information which needs to be inserted as follows:

```
# Example Vehicle: the following data has been input from the vehicle
# inventory Id of 24444
SGR_Prediction = [
      25, # Input the length of the Vehicle
```

```
220515, # Input the Average Lifetime Miles per Active Vehicles
22380, # Input the Total Miles on Active Vehicles During Period
16, # Input the Seating Capacity of the vehicle
0, # Input the Standing Capacity of the vehicle
2, # Input ADA Fleet Vehicles
2, # Input the Active Fleet Vehicles
0, # Input the Emergency Contingency Vehicles
2, # Input Total Fleet Vehicle
16, # Input the "number" based on Fuel Type of vehicle
11, # Input the "number" based on Mode of vehicle
14, # Input the "number" based on Vehicle Type of vehicle
1.5625, # Input the ratio of "Vehicle length" and "Seating
      Capacity of the vehicle"
0, # Input the ratio of "Rebuild Year" and "Manufacture Year"
0, # Input the ratio of "length of the Vehicle" and "Standing
      Capacity of the vehicle"
0, # Input Ratio of "Standing Capacity of the vehicle" and
      "Seating Capacity of the vehicle"
11190.0, # Input Ratio of "Total Miles on Active Vehicles During
      Period" and "Total Fleet Vehicles"
11190.0, # Input Ratio of "Total Miles on Active Vehicles During
      Period" and "Active Fleet Vehicles"
110257.5, # Input Ratio of "Average Lifetime Miles per Active
      Vehicles" and "Total Fleet Vehicles"
441030.0, # Input Ratio of "Average Lifetime Miles per Active
      Vehicles" and "Active Fleet Vehicles"
0, # TOS: if TOS = DO, then input 1; else input 0
1, # TOS: if TOS = PT, then input 1; else input 0
False, # Mode: if Mode = CC, input True; else input False
False, # Support Mode: if Support Mode = MB, input True; else
      input False
True, # Fuel Type: if Fuel Type = Diesel Fuel, input True; else
      input False
True, # Vehicle Type: if Vehicle Type = Bus, input True; else
      input False
True, # Ownership Type: if Ownership Type = OOPA, input True;
      else input False
```

```
        False, # Funding Source: If Funding Source = NFPA, input True;
              else input False
        False, # Funding Source: if Funding Source = OF, input True; else
              input False
        True # Funding Source: if Funding Source = UA, input True; else
              input False
        ]
```

The following blocks of codes generates the prediction for a single vehicle:

```
# To predict the service life of a single vehicle
vehicle_service_life = [SGR_Prediction]
# Run the model and make a prediction for each vehicle
predicted_service_life = model.predict(vehicle_service_life)
# Predicting the single vehicle
predicted_life = predicted_service_life[0]
```

The following code prints the value of the predicted service life of the single vehicle:

```
# Predict the service life for the vehicle
print ("The predicted service life of the vehicle would be {:, .0f}
      years".format(predicted_life))
```

And, the output is as follows:

```
  The predicted service life of the vehicle would be 12 years
```

Finally, the following block of code calculates the predicted retired year of the vehicle:

```
# Add Predicted Service Life to the non-retired revenue vehicle
# inventory data with only specific vehicle
revenue_single_vehicle = revenue_all[revenue_all['Revenue Vehicle
      Inventory ID'] == 24444]
revenue_single_vehicle['Predicted Service Life'] = predicted_life
# Create a new column by adding predicted service life with Manufacture
  Year
revenue_single_vehicle['Projected Retired Year'] =
      (revenue_single_vehicle['Manufacture Year'] +
      (revenue_single_vehicle ['Rebuild Year'] -
      revenue_single_vehicle['Manufacture Year']).fillna +
      revenue_single_vehicle['Predicted Service Life'] + 1.0).round()
```

```
# Print the result
revenue_single_vehicle
```

The above code printed the output of the vehicle with the predicted retired year. Since there were 29 columns in the output, a few important columns were inserted for simplicity shown in Table 44. The gradient boosting regression predictive model with top the 30 important features predicts that the vehicle with RVI ID of 24444 should be retired in the year of 2019. This model predicted the same projected retired year comparing with the prediction of the same vehicle made by the gradient boosting regression predictive model with all features.

Table 44. The Predicted Retired Year for the Vehicle with RVI ID of 24444

Revenue Vehicle Inventory ID	Agency Name	Fuel Type	Man. Year	…	Vehicle Length	Vehicle Type	Projected Retired Year
24444.0	City of Fargo, DBA: Metropolitan Area Transit	Diesel	2006	…	True	Bus	2019

4.20. Challenges

Throughout the preprocessing of revenue vehicle inventory data for machine learning algorithms and exploratory data analysis, many challenges were encountered. For instance, the quality of the revenue vehicle inventory data was not good. In addition, there were many roadblocks during the feature engineering such as problems with missing data.

Data are the most important part of developing any predictive model. Lack of good quality data or lack of sufficient data may not produce a good predictive model. In this model, the revenue vehicle inventory data from 2008 to 2016 from the NTD database were used. Due to poor quality of data, the available data from 1999 to 2007 were not used in the model. According to the FTA, the vehicle's default useful life depends on the vehicle type (NTD, 2017). Therefore,

each vehicle type needs enough training data to train the model. The exploratory data analysis with the training data showed some of the vehicle types only had a few training data points. For example, the Inclined Plane Vehicle and Double Decker Bus vehicle types had only 1 data point, which was not enough to train these particular vehicle types.

The tasks for data preparation of the machine learning algorithm were very challenging. The tasks involved cleaning bad data with missing information, creating new features, transforming them into useful features, and reorganizing data into suitable machine learning algorithms. The data preparation involved looking for data anomalies and making sure to fix anomalies by taking proper actions and transforming them to be consistent.

Since the revenue vehicle inventory data sets were complex and there was no direct information on when a vehicle was retired, it was very challenging to split the data into the training set with retired vehicles and the deployment set with non-retired vehicles. In the revenue vehicle inventory data, the Retired column was an important attribute as it indicated whether a vehicle was retired or not by flagging 'Y' or 'N'. This column exists in the data from 2014 through 2016, but not in the data from 2013 and prior. In addition, there were many data points where the Retired column had null points in data from 2014 through 2016. Therefore, during the data cleaning process, the Retired column was added to data from 2008 to 2013 with 'Y' value and an extra column Retired Year was created to all data sets.

The Manufacture Year was another important column used to calculate the service life of vehicles. There were 3189 data points with no value for Manufacture Year, which represented about 7.5% of the total data. These data were not considered for the predictive model and removed from the data set. Fuel Type was also an important categorical feature that impacted the accuracy of the predictive model. The exploratory analysis showed that there were 14100 data

points missing for the Fuel Type category, which represented 33% of total data points. However, in this case, the huge amount of data was not dropped from the data set. Instead, the missing category was replaced by a dummy type with Unknown Fuel type. It might impact the performance; however, it solved the problem.

During data processing, creating some useful features by combining multiple features was another challenge. Since there was no strong correlation found between features with the target feature, a combination of different features was applied to the model to obtain the best performance. Therefore, a trial and error method was applied to the features selection using the feature importance function to see whether newly created features had any impact on the model. By following the trial and error method, some of the features were selected for the model, and the rest of them were rejected.

After completing the initial exploratory data analysis, the selection of the best predictive model for this problem was another challenge. The analysis showed the target variable was a continuous variable and the regression analysis could solve the problem. Since there are many regression algorithms available for machine learning problems and there is no concrete methodology to choose the best model, this work was started with several popular methods to build the predictive model for this problem. The entire data set was split into three sets called the training set, the test set, and the deployment set. Once the process was done, three popular machine learning techniques were chosen for the model. They were random forest regression, gradient boosting regression, and decision tree regression. By using these three techniques, a separate predictive model was built, evaluated the performance of the results, and the performance results were compared across models. Even though the evaluation and the

comparison of the models took a significant amount of time, choosing the perfect algorithm for the problem was a bit of a challenge.

Another challenge was to handle the outliers in the data set. After calculating the actual service life of the vehicle by subtracting the manufacture year from the retired year some vehicles were observed to have very low service life. This may be due to some consequences of human errors by incorrectly inputting data for manufacture year or retired during the data collection processes. These data were handled by removal from the training data set.

4.21. Summary of Data Analysis and Results

This chapter explored revenue vehicle inventory data set from the NTD database where transit agencies publish their vehicle information at the end of each fiscal year. Python was used as a programming language in the Jupyter Notebook environment to analyze the revenue vehicle inventory data and develop the predictive model. Nine data sets were used, one from each year between 2008 to 2016; however, they were not consistent because data varied between years. Therefore, each data set needed to be cleaned up individually and be made consistent before combining them. A new column Retired Year was added and calculated the value based on the status of the vehicle's retirement. There were many data issues in the initial combined data set. One of the main issues was missing information. The missing information was handled by filling missing values with either zero (0) or by applying a function. In some cases, the data points with missing values were removed. The categorical names with missing values were filled with 'Unknown' as the keyword followed by underscore, and then category name. Finally, some unnecessary variables were removed from the data set as they were redundant for the model.

After cleaning up the data, a new column Service Life was created, and values were generated by subtracting Manufacture Year from Retire Year. The entire data set was split into

two sets. The retired vehicles were used as the training set, while the non-retired vehicles were used as the deployment set. An exploratory data analysis was performed on the training set to see the significance value of data in the model as well as visualize outliers, data distribution, and relationships between features.

The features of the training set were engineered prior to building a machine learning model. As part of the feature engineering process, several new features were created by combining different numerical features. In addition, binary features were created from categorical names, and a few additional features were created by analyzing the histogram from categorical features. A deployment set was created in the same manner. A model was built from the training set, then it was applied to the deployment set for predictions.

Since the training data had the target variable, it was used to train the model. Three different machine learning algorithms called random forest regression, gradient boosting regression, and decision tree regression were applied to build three different predictive models. Before building the model, the parameters for each algorithm were tuned to optimize the performance of the model. During modeling, the training data was split into the training set and the test set in the ratio of 70% of data to train the model and 30% of data to evaluate the model. As part of the evaluation, three performance metrics called root mean squared error, mean absolute error, and R^2 score were applied to see how accurately the models were performing. After comparing the performance results, the gradient boosting regression predictive model was selected because it provided better performance results for the problem.

Sometimes, a large number of features may cause problems to generalize a model. Therefore, the feature importance ranking method was further applied to the gradient boosting regression model to get the top 30 most important features. After applying the top 30 most

important features and comparing the performance of the previous gradient boosting regression model, we found an even better performing predictive model. Finally, we applied the full data set as a training set to train the model that further improved the performance of the predictive model. We concluded the gradient boosting regression model using the full training data set with the top 30 most important features would be our final predictive model.

After developing the predictive model using the gradient boosting regression algorithm with the top 30 most important features, the model was applied on the deployment set for predictions. Results were saved in a CSV file for transit agencies, and further data analysis was performed to visualize the current conditions of the nation's transit vehicles. Special data analysis was performed on the Fargo Metropolitan Area Transit (MAT Bus) agency as an example of how transit agencies could perform their own data analysis. By conducting similar analysis, transit agencies would be made aware of their vehicle conditions to determine replacement and rehabilitation needs.

Finally, another supplementary model was built using the gradient boosting regression model with the top 30 most important features to make predictions for any given vehicle. Using this simple model, transit agencies could input the necessary vehicle information for a specific vehicle in the model and predict the condition of the vehicle.

CHAPTER 5. CONCLUSION AND FURTHER RESEARCH

5.1. Conclusion

This research developed a predictive model to evaluate the transit state of good repair using machine learning algorithms. This dissertation explored three machine learning techniques to predict the service life of vehicles. The random forest regression, gradient boosting regression, and decision tree regression were applied on revenue vehicle inventory data to build the predictive model and predict vehicle service life. After evaluating and comparing performance results, we found that the gradient boosting regression predictive model performed better than the other two predictive models. The gradient boosting regression algorithm was also used to identify the top most important features, and the predictive model with the top 30 most important features worked even better to predict vehicle service life.

The predictive tool developed in this study allows transit agencies to predict the service life of their revenue vehicles. Furthermore, the FTA can use this tool to see the overall condition of the nation's revenue vehicles. Even though, the performance of the predictive tool is very good, it could be further improved by implementing the following recommendations.

5.2. Recommendation

The author recommends to add additional data to the training set to train the model in future work. The other data can be found from the FTA or directly from the transit agencies. The analysis showed that if more training data can be added to the predictive model, the performance of the model will be improved. The FTA can also take an initiative to add few crucial columns in the revenue vehicles inventory database. For example, the FTA can instruct transit agencies to add 'operating start date,' 'retired date,' 'cost of vehicles,' and 'agency zone' columns in the database. The above information will improve the predictive performance for the model.

The exploratory data analysis also showed that some extreme values in the data were causing outliers in the data. For example, in some cases, the retired year was earlier than the manufacture year which was creating negative service life of vehicles. The author recommends that the FTA will take actions to improve the quality of the revenue vehicle inventory data by correcting manufacture year or retired year in the NTD database. The author suggests that further analysis of revenue vehicle inventory data should be an essential step to solve the issues in the state of good repair.

5.3. Further Research

The author suggests that the research of machine learning algorithms on the state of good repair problem has enormous potential for further analysis. Adding few features as suggested earlier and selecting better features for the model may produce perfect results. Therefore, feature engineering can be further processed by combining different features and further research can be done to choose various features selection processes.

In this study, the backlog analysis was not done as cost related data were not available in the NTD database. However, the cost of the revenue vehicles can be collected by doing further research. Therefore, in future, the backlog analysis can be added to the method, and the backlog can be estimated to maintain the state of good repair. Further improvement of this predictive model will help transit agencies to predict the service life of their vehicles very well so that the agency can plan and prioritize to replace or rehabilitate their assets accordingly.

REFERENCES

Amtrak. (2009). *Northeast Corridor State of Good Repair Spend Plan.* Washington, DC.: Amtrak.

APTA. (2007). *Public transportation: Benefits for the 21st century.* Washington, D.C.: American Public Transportation Association. Retrieved from Public Transportation: http://www.apta.com/resources/reportsandpublications/Documents/twenty_first_century.pdf

APTA. (2013). *Defining a Transit Asset Management Framework to Achieve a State of Good Repair: Recommended Practice.* Washington, D.C.: APTA Standards Development Program Working Group: Transit Asset Management.

APTA. (2013b). *Creating a Transit Asset Management Program: Recommended Practice.* Washington, DC: American Public Transportation Association, Working Group: Transit Asset Management.

APTA. (2016). *A Guide to Public Transportation and Rail-Related Provisions: Fixing America's Surface Transportation Act (FAST ACT).* Washington, D.C.: American Public Transportation Association.

APTA. (2017). *Transit Facts: Transit Lifestyle. Public Transportation*. Washington, D.C.: American Public Transportation Association. Retrieved from Public Transportation: http://www.publictransportation.org/benefits/Pages/Transit-Lifestyle.aspx

Aurlien, G. (2017). *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* O'Reilly Media, Inc.

Bowles, M. (2015). *Machine Learning in Python: Essential Techniques for Predictive Analysis.* Indianapolis, IN: John Wiley & Sons, Inc.

Brownlee, J. (2013). A tour of machine learning algorithms: Machine Learning Mastery.

BTS. (2016). *Transportation Statistics Annual Report.* U.S. Department of Transportation. Washington, D.C.: Bureau of Transportation Statistics.

Cambridge Systematics. (2005). *Analytical tools for Asset Management.* Washington, D.C.: Transportation Research Board.

Cambridge Systematics. (2006). *Performance measures and targets for transportation asset management.* Washington, D.C.: Transportation Research Board.

Cambridge Systematics. (2009). *Virginia's Long-Range Multimodal Transportation Plan 2007-2035.* Office of Intermodal Planning and Investment. VTrans.

Cevallos, F. (2016). *State of Good Repair Performance Measures: Assessing Asset Condition, Age, and Performance Data.* National Center for Transit Research.

Cohen, H., & Barr, J. (2012). *State of Good Repair: Prioritizing the Rehabilitation and Replacement of Existing Capital Assets and Evaluating the Implications for Transit.* Transportation Research Board.

Contingency table. (2018). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Contingency_table

Downey, A. B. (2014). *Think stats: exploratory data analysis.* O'Reilly Media, Inc.

Edrington, S., Brooks, J., Cherrington, L., Hamilton, P., Hansen, T., Pourteau, C., & Sandidge, M. (2014). *Guidebook: Managing Operating Costs for Rural and Small Urban Public Transit Systems.* Texas A&M Transportation Institute.

FHWA. (2010). *Data Integration Primer.* Office of Asset Management, U.S. Department of Transportation. Washington, D.C.: Federal Highway Administration.

FTA. (2008). *Transit State of Good Repair: Beginning the Dialogue.* U.S. Department of Transportation. Washington, D.C: Federal Transit Administration.

FTA. (2010a). *National State of Good Repair Assessment.* U.S. Department of Transportation. Washington, D.C.: Federal Transit Administration.

FTA. (2010b). *Transit Asset Management Practices: A National and International Review.* U.S. Department of Transportation. Washington, D.C.: Federal Transit Administration.

FTA. (2011). *State of Good Repair Initiative Report to Congress.* U.S. Department of Transportation. Washington, D.C.: Federal Transit Administration.

FTA. (2012). *MAP-21 Fact Sheet: State of Good Repair Grants.* U.S. Department of Transportation. Washington, D.C.: Federal Transit Administration.

FTA. (2013). *Transit Economic Requirements Model.* U.S. Department of Transportation. Washington, D.C.: Federal Transit Administration.

FTA. (2017, February 22). *FAST Act: State of Good Repair*. (F. T. Administration, Editor) Retrieved 2017, from U.S. Department of Transportation: https://www.transit.dot.gov/FAST

FTA. (2017b). *National Transit Database: 2017-2018 Asset Inventory Module Reporting Guide.* US Department of Transportation. Washington, D.C.: Federal Transit Administration. Retrieved from FTA Office of Budget and Policy.

FTA. (2017c). *National Transit Database: Asset Inventory Module.* Washington, D.C.: Federal Transit Administration.

FTA. (2017d). National Transit Database: What is the National Transit Database (NTD) Program? Washington, D.C.: Federal Transit Administration. Retrieved 2018, from Federal Transit Administration: https://www.transit.dot.gov/ntd/what-national-transit-database-ntd-program

Gagne, D. J., McGovern, A., Haupt, S. E., & Williams, J. K. (2017). *Evaluation of statistical learning configurations for gridded solar irradiance forecasting*. Solar Energy *150*, 383-393.

Garreta, R., & Moncecchi, G. (2013). *Learning scikit-learn: Machine Learning in Python.* Birmingham, UK: Packt Publishing Ltd.

Geitgey, A. (2017). *Machine Learning & AI Foundations: Value Estimations.* Lynda.com. Lynda. Retrieved from https://www.lynda.com/Data-Science-tutorials/Machine-Learning-Essential-Training-Value-Estimations/548594-2.html

Grus, J. (2015). *Data science from scratch: first principles with python.* (Vol. First Edition). (M. Beaugureau, Ed.) Sebastopol, CA, USA: O'Reilly Media, Inc.

Hunter, J., Dale, D., Firing, E., & Droettboom, M. (2017). The Matplotlib 2.1.0, User's Guide. *Matplotlib*. Retrieved from https://matplotlib.org/2.1.0/users/index.html

Inyang, F. I., Ozuomba, S., & Ezenkwu, C. P. (2017). *Comparative analysis of Mechanisms for Categorization and Moderation of User Generated Text Contents on a Social E-Governance Forum*. Mathematical and Software Engineering, 78-86.

Jain, A. (2016). *Complete Guide to Parameter Tuning in Gradient Boosting (GBM) in Python.* Retrieved 2017, from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/

Johnson, N. E., Ianiuk, O., Cazap, D., Liu, L., Starobin, D., Dobler, G., & Ghandehari, M. (2017). *Patterns of waste generation: A gradient boosting model for short-term waste prediction in New York City.* Waste Management, *V 62*, 3-11. doi:10.1016/j.wasman.2017.01.037

Jordan, M., & Mitchell, T. (2015, July 17). *Machine Learning: Trends, perspectives, and prospects*. Science, *349*(6245), 255-260. doi:10.1126/science.aaa8415

Kumar, A. (2016). *Learning Predictive Analytics with Python* (Vol. First Edition). (N. Amey, Ed.) Birmingham, UK: Packt Publishing Ltd.

Lauren, I., & Rose, D. (2012). Transit Asset Management Manual - Overview. *4th State of Good Repair Roundtable.* Philadelphia, PA: Federal Transit Administration.

Laver, R., Schneck, D., Skorupski, D., & Cham, L. (2007). *Useful life of transit buses and vans.* U.S. Department of Transportation. Washington, D.C.: Federal Transit Administration.

Lee, Y.-J., & Min, O. (2017). Comparative Analysis of Machine Learning Algorithms to Urban Traffic Prediction. *Information and Communication Technology Convergence (ICTC).* IEEE. doi:10.1109/ICTC.2017.8190846

Louch, H., Robert, W., Gurenich, D., & Hoffman, J. (2009). *Asset Management Implementation Strategy.* Washington, D.C.: New Jersey Department of Transportation. Retrieved from http://www.state.nj.us/transportation/refdata/research/reports/NJ-2009-005.pdf

Ma, J. (2012). Parameter tuning using gaussian processes. *Ph.D. Dissertation*. Hamilton, New Zealand. Retrieved from https://hdl.handle.net/10289/6497

Massaron, L., & Boschetti, A. (2016). *Regression Analysis with Python.* (Vol. First). Birmingham, UK: Packt Publishing Ltd.

McCollom, B. E., & Berrang, S. A. (2011). *Transit Asset Condition Reporting: A Synthesis of Transit Practice.* Transportation Research Board of the National Academies. Washington, D.C.: Federal Transit Administration.

McKinney, W. (2017). *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython.* (2nd ed.). Sebastopol, CA, USA: O'Reilly Media, Inc.

McKinney, Wes; PyData Development Team. (2017). *API Reference - pandas 0.22.0 documentation: powerful Python data analysis toolkit Release 0.22.0.* Pandas. Retrieved from https://pandas.pydata.org/pandas-docs/stable/

Meyer, M. D., & Cambridge Systematics, Inc. (2007). *US Domestic Scan Program: Best Practices in Transportation Asset Management.* Federal Highway Administration; National Cooperative Highway Research Program; The American Association of State Highway and Transportation Officials.

Mirjalili, V., & Raschka, S. (2017). *Python Machine Learning. Second Edition.* (2nd ed.). Birmingham, UK: Packt Publishing Ltd.

MTA. (2014). *A Bold Direction for Leading Transportation in the Next 100 Years.* New York, NY: Metropolitan Transportation Authority (MTA). Retrieved from http://web.mta.info/mta/news/hearings/pdf/MTA_Reinvention_Report_141125.pdf

Mueller, J. P., & Massaron, L. (2015). *Python for Data Science for Dummies.* Hoboken, New Jersey, USA: John Wiley & Sons, Inc.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. (M. Braun, Ed.) *Journal of Machine Learning Research, 12*, 2825-2830.

Raschka, S. (2015). *Python machine learning* (First ed.). (R. Banerjee, Ed.) Birmingham, UK: Packt Publishing Ltd.

Rathore, S. S., & Kumar, S. (2016). A decision tree regression based approach for the number of software faults prediction. *ACM SIGSOFT Software Engineering Notes, v. 41*(no. 1), 1-6.

Robert, W., Reeder, V., Lawren, K., Cohen, H., & O'Neil, K. (2014). *Guidance for Developing a Transit Asset Management Plan.* Transportation Research Board. Washington, D.C.: Federal Transit Administration in cooperation with the Transit Development Corporation. doi:10.17226/22306

Robert, William; Reeder, Virginia; Lauren, Katherine. (2014). *Guidance for Developing the State of Good Repair Prioritization Framework and Tools.* Transit Cooperative Research Program. Arlington, MA: Spy Pond Partners, LLC.

Rose, D., Lauren, I., Shah, K., Blake, T., & Parsons Brinckerhoff, I. (2012). *Asset Management Guide: Focusing on the Management of Our Transit Investments.* U.S. Department of Transportation. Washington, D.C.: Federal Transit Administration. Retrieved from https://www.transit.dot.gov/about/research

Shen, Q., & Chouchoulas, A. (2001). Rough set-based dimensionality reduction for supervised and unsupervised learning. *Applie Mathematics and Computer Science 11 No. 3*, 583-602.

Springstead, D. (2011). Asset Management: An Agency Perspective. *90th Annual Meeting of the Transportation Research Board.* Washington, D.C.: Transportation Research Board.

TRB. (2013). *Review of the Federal Transit Administration's Transit Economic Requirements Model.* Transportation Research Board. Washington, D.C.: National Research Council.

US Congress. (2012). *Moving Ahead for Progress in the 21st Century (MAP-21).* 112th Congress Public Law, Washington, D.C.

US DOT. (2013). *Transportation for a New Generation: Strategic Plan.* Washington, D.C.: US Department of Transportation.

US GAO. (2013). *Transit Asset Management: Additional Research on Capital Investment Effects Could Help Transit Agencies Optimize Funding.* U.S. Government Accountability Office. Retrieved from United States Government Accountability Office (U.S. GAO): www.gao.gov/products/gao-13 - 571

VDOT. (2006). *Asset Management Methodology (Appropriation Act Item 444 A).* Virginia: Virginia Department of Transportation.

Waaramaa, E. R. (2010). *Asset Management Systems MBTA Approach and Lessons Learned.* Massachusetts Bay Transportation Authority. Chicago, IL: State of Good Repair Roundtable.

Welbes, M. J. (2009). *Rail Modernization Study.* U.S. Department of Transportation. Washington, D.C.: Federal Transit Administration.

Xu, M., Watanachaturaporn, P., Varshney, P. K., & Arora, M. K. (2005). Decision tree regression for soft classification of remote sensing data. *Remote Sensing of Environment, 97*(3), 322-336.

Zarembski, A. M. (2013). *Analysis of Transit 20 Year Capital Forecasts: FTA TERM Model vs. Transit Estimates.* Washington, D.C.: Transportation Research Board of the National Academies. Retrieved from http://onlinepubs.trb.org/onlinepubs/reports/TERM_March_2013Zarembski.pdf

Zhang, Y., & Haghani, A. (2015). A gradient boosting method to improve travel time prediction. *Transportation Research Part C: Emerging Technologies*, 308-324.

Zhao, Y., & Zhang, Y. (2008). Comparison of decision tree methods for finding active objects. *Advances in Space Research, 41*(no. 12), 1955-1959.

Zhou, Z.-H. (2012). *Ensemble Methods: Foundations and Algorithms.* (R. Herbrich, & T. Graepel, Eds.) Boca Raton, FL, USA: Chapman & Hall/CRC.

APPENDIX A. FIELDS IN THE REVENUE VEHICLE INVENTORY MODULE

Transit agencies group together vehicles if they are identical in manufacture, vehicle type, vehicle mode, and funding source. These identical vehicles are called fleets, and they are reported to NTD database. Transit agencies collect the following revenue vehicle information and report to NTD database (FTA, 2017b; FTA, 2017c):

Agency Fleet Identification – The vehicle identification is the unique number provided by the FTA. Transit agencies must report each fleet with the unique identifier to the inventory.

Mode – Transit agencies need to report the primary mode of each fleet to inventory.

Vehicle Type – Transit agencies need to report the type of revenue vehicle for each fleet.

Total Fleet – Total fleet includes the number of vehicles in both active and inactive fleets. Transit agencies need to report the number of vehicles in the total fleet at the end of fiscal year.

Number of Active Vehicles in Fleet – These are the vehicles which are still active at the fiscal year-end. Agencies report the number of active vehicles in the fleet at the end of the year.

Dedicated Fleet – Dedicated fleets are vehicles which are dedicated to only used for public transportation services. Agencies need to report directly operated vehicles under dedicated fleet.

Vehicle Length – Vehicle length is the length in feet for each fleet of vehicles. Transit agencies should report it to inventory.

Seating Capacity – Manufacturer cites the number of seating capacity for the vehicle. Transit agencies need to report the actual number to the inventory.

Standing Capacity – Standing capacity is the maximum number of people who are allowed to stand inside on the vehicle. Transit agencies should report the number of standing

capacity. If the policy does not allow people to stand on the vehicle, they should report zero for standing capacity.

Year of Manufacture – This is the year of manufacture when the vehicle was originally built. Transit agencies must need to report the year to inventory.

Ownership – Agencies need to report what entity owns the vehicles and the ownership type.

Funding Source – There are several funding sources available to purchase or lease vehicles. Transit agencies must need to report the funding sources.

Number of Emergency Contingency Vehicles – Transit agencies may keep the FTA funded vehicles in an inactive fleet if they are used in case of natural disasters. The agencies need to report the number of emergency vehicles as an inactive fleet.

ADA Accessible Vehicles – These are the active vehicles that meet accessibility requirements of Americans with Disability Act of 1990 (ADA). Transit agencies need to report the number of ADA vehicles.

Fuel Type – Transit agencies need to report the type of fuel used to operate the revenue vehicles.

Year of Rebuild – Transit agencies must report the year of the rebuild if it is rebuilt. Under the FTA grant rules, if the bus is rebuilt, the service life will be extended to a minimum of four years, and if a rail vehicle is rebuilt, the service life will be extended to a minimum of 10 years.

Manufacturer – Transit agencies need to report the manufacturer of the vehicle or the final manufacturer of the vehicle if more than one manufacturer.

Model – Transit agencies need to report the model of the vehicle that manufacturer provides.

Total Miles on Active Vehicles - Transit agencies need to report total miles on active vehicles during the fiscal year.

Average Lifetime Mileage per Active Vehicle – It is the average mileage which begins with the original manufacturer data. Transit agencies need to report it at the end of fiscal year.

Support Other Mode – If active vehicles are used to provide on two modes, transit agencies need to report the supports another mode for active vehicles which provide service for another mode.

APPENDIX B. NTD REVENUE VEHICLE INVENTORY MODULE

In this research, the Revenue Vehicle Inventory data from National Transit Database had been used. The transit agencies reported information of revenue vehicles at the end of the fiscal year in the revenue inventory repository. The revenue information data are shown in Table A1.

Table B1. Revenue Vehicles

Revenue Vehicles Fields
5 Digit NTD ID
Legacy NTD ID
Agency Name
Reporter Type
Reporting Module
Mode
TOS
Revenue Vehicle Inventory ID
Total Fleet Vehicles
Dedicated Fleet
Vehicle Type
Ownership Type
Funding Source
Manufacture Year
Rebuild Year
Manufacturer
Other Manufacturer
Description
Model
Active Fleet Vehicles
ADA Fleet Vehicles
Emergency Contingency Vehicles
Fuel Type
Vehicle Length
Seating Capacity
Standing Capacity
Total Miles on Active Vehicles During Period
Average Lifetime Miles per Active Vehicles
Supports Mode
Supports Service
Retired

Source: Adapted from Federal Transit Administration. 2017. *National Transit Database: NTD Policy Manual.* Office of Budget and Policy. U.S. Department of Transportation. Washington, D.C.

Table B2. Vehicle Type

Vehicle Code	Vehicle Type	Vehicle Code	Vehicle Type
AB	Articulated bus	LR	Light rail vehicle
AG	Automated guideway vehicle	MO	Monorail vehicle
AO	Automobile	RL	Commuter rail locomotive
BR	Over-the-road bus	RP	Commuter rail passenger coach
BU	Bus	RS	Commuter rail, self-propelled pass car
CC	Cable car	SB	School bus
CU	Cutaway bus	SV	Sports Utility Vehicle
DB	Double decked bus	TB	Trolleybus
FB	Ferryboat	TR	Aerial tramway
HR	Heavy rail passenger car	VN	Van
IP	Inclined plane vehicle	VT	Vintage trolley/streetcar

Source: Adapted from Federal Transit Administration. 2017. *National Transit Database: NTD Policy Manual*. Office of Budget and Policy. U.S. Department of Transportation. Washington, D.C

Table B3. Fuel Type

Fuel Code	Fuel Type	Fuel Code	Fuel Type
BD	Bio-diesel	GA	Gasoline
BF	Bunker fuel	HD	Hybrid diesel
CN	Compressed natural gas (CNG)	HG	Hybrid gasoline
DF	Diesel fuel	HY	Hydrogen
DU	Dual fuel	KE	Kerosene
EB	Electric battery	LN	Liquefied natural gas (LNG)
EP	Electric propulsion	LP	Liquefied petroleum gas (LPG)
ET	Ethanol	MT	Methanol

Source: Adapted from Federal Transit Administration. 2017. *National Transit Database: NTD Policy Manual*. Office of Budget and Policy. U.S. Department of Transportation. Washington, D.C

Table B4. Funding Source

Funding Code	Funding Source	Funding Code	Funding Source
UA	Urbanized Area Formula Program	NFPE	Non-Federal private funds
OF	Other Federal funds	RAFP	Rural Area Formula Program
NFPA	Non-Federal public funds	EMSID	Enhanced Mobility for Seniors and Individuals with Disabilities

Source: Adapted from Federal Transit Administration. 2017. *National Transit Database: NTD Policy Manual.* Office of Budget and Policy. U.S. Department of Transportation. Washington, D.C

Table B5. Ownership Type

Ownership Code	Ownership Type	Ownership Code	Ownership Type
LPPA	Leased under lease purchase agreement by a public agency	OOPA	Owned outright by public agency
LPPE	Leased under lease purchase agreement by a private entity	OOPE	Owned outright by private entity
LRPA	Leased or borrowed from related parties by a public agency	TLPA	True lease by a public agency
LRPE	Leased or borrowed from related parties by a private entity	TLPE	True lease by a private entity

Source: Adapted from Federal Transit Administration. 2017. *National Transit Database: NTD Policy Manual.* Office of Budget and Policy. U.S. Department of Transportation. Washington, D.C

Table B6. Vehicle Mode

Mode Code	Primary Mode	Mode Code	Primary Mode
AG	Automated Guideway	LR	Light Rail
AR	Alaska Railroad	MB	Bus
CB	Commuter Bus	MG	Monorail/Automated Guideway
CC	Cable Car	PB	Public Bus
CR	Commuter Rail	RB	Bus Rapid Transit
DR	Demand Response	SR	Streetcar Rail
FB	Ferry Boat	TB	Trolleybus
HR	Heavy Rail	TR	Aerial Tramway
IP	Inclined Plane	VP	Vanpool
JT	Jitney	YR	Hybrid Rail

Source: Adapted from Fabian Cevallos. 2016. State of Good Repair Performance Measures: Assessing Asset Condition, Age, and Performance Data. Final Report for National Center for Transit Research (NCTR) and University of South Florida.